PROGRAMMER TO PROGRAMMER™

Professional CSS
Cascading Style Sheets for Web Design, 2nd Edition

CSS Web设计高级教程（第2版）

(美) Christopher Schmitt
Todd Dominey
等著
窦朝晖
译

wrox™
WILEY
清华大学出版社

CSS Web
设计高级教程(第2版)

(美) Christopher Schmitt
Todd Dominey 等著
窦朝晖 译

清华大学出版社
北京

大家网[www.topsage.com]

最新 JavaScript、Ajax 典藏级学习资料下载分类汇总

JavaScript 初学者及参考必备：

JavaScript 学习指南（第 2 版）

JavaScript: The Definitive Guide, 6th Edition (JavaScript 权威指南 第 6 版)PDF+epub

JavaScript 权威指南 (第 5 版) 中文版 | 英文版+随书源代码 | 第 4 版 英文版 |

JavaScript 高级程序设计 (第 2 版) 中文版

JavaScript Bible, 7th Edition (JavaScript 宝典 第 7 版)

JavaScript 宝典 (第 6 版) 中文版 | 英文版

JavaScript 入门经典（第 3 版）中文高清 PDF 下载

JavaScript 语言精粹 高清 PDF 中文版 | 英文 chm 版 | 英文 pdf 版

JavaScript 开发技术大全

JavaScript & DHTML Cookbook 中文版（第 2 版）

JavaScript 捷径教程

JavaScript 实战 (Practical JavaScript, DOM Scripting, and Ajax Projects) 中文版

JavaScript DOM 高级程序设计 (中文版高清 PDF 下载)

ppk 谈 JavaScript (中文高清 PDF)

JavaScript 设计模式 (Pro JavaScript Design Patterns) 中文版

JavaScript 模式

JavaScript 王者归来

Javascript Bible Golden Edition (JavaScript 金典)

JavaScript The Complete Reference (JavaScript 技术大全)

Advanced Javascript, 3rd Edition (JavaScript 高级编程)

JavaScript Examples Bible (JavaScript 实例宝典)

Wrox Beginning JavaScript (第三版)

Professional JavaScript for Web Developers

O'Reilly Head First Javascript

Pro Javascript RIA Techniques: Best Practices, Performance and Presentation

JScript 中文参考手册

Javascript 程序员字典

Special Edition Using JavaScript

Object Oriented JavaScript

O'Reilly JavaScript Patterns

JavaScript DOM 编程艺术第一版中英文 | 第二版英文

JavaScript 与 Jscript 从入门到精通

The.Pragmatic.Bookshelf 开发丛书-JavaScript 实用指南.Pragmatic.Guide.to.JavaScript.Christophe.Porteneuve.文字版

精通 JavaScript(图灵计算机科学丛书)

程序员常用 JavaScript 特效

ASP .NET 2.0 Demystified

w3school.Javascript 特效大全(上册)

High Performance JavaScript（中英文对照版）

CSS&javascript 动态网页设计与制作

JavaScript 使用手册.rar

JavaScript 网页特效范例宝典 | 源码

实用 JavaScript 网页特效编程百宝箱

JavaScript 客户端验证和页面特效制作

JavaScript 框架(JavaScript/Ajax Frameworks)：

jQuery 基础教程（第 2 版）中文高清 PDF 下载 | 英文版

jQuery 实战（jQuery in Action）中文高清 PDF 下载 | 英文版

锋利的 jQuery

jQuery 基础教程 中文高清 PDF 版

jQuery 攻略 (jQuery Recipes: A Problem Solution Approach) 中文 PDF | 英文版

15 天学会 jQuery (PDF 中文版)

O'Reilly jQuery Pocket Reference

jQuery Reference Guide

jQuery 1.4 Plugin Development Beginner's Guide

jQuery 1.2 API 全中文

jQuery 中英文对照手册

jQuery: Visual QuickStart Guide

Ext JS in Action

Prototype and Scriptaculous in Action

Prototype and script.aculo.us

精通 Dojo 中文版 PDF

Dojo: The Definitive Guide — Dojo 权威指南

Mastering Dojo: JavaScript and Ajax Tools for Great Web Experiences

Apress Practical Dojo Projects

Dojo: Using the Dojo JavaScript Library to Build Ajax Applications

Mastering Dojo: JavaScript and Ajax Tools for Great Web Experiences

Apress MooTools Essentials

Beginning Google Web Toolkit: From Novice to Professional

jQuery 开发视频教程 jQuery Projects: Creating an Interactive Photo Gallery

Pro Android Web Apps: Develop for Android using HTML5, CSS3 & JavaScript

Building.iPhone.Apps.with.HTML,CSS,and.JavaScript

AJAX (Asynchronous JavaScript and XML)：

AJAX 完全手册 (AJAX: The Complete Reference) 中文版 PDF 下载

Wiley AJAX Bible (Ajax 宝典)

MANNING AJAX In Action

Ajax 基础教程

XMLHttpRequest 中文参考手册

征服 Ajax Web 2.0 开发详解

Wrox Beginning Ajax

Wrox Professional Ajax, 2nd Edition (Ajax 高级编程)

Wrox Professional Rich Internet Applications AJAX and Beyond

O'Reilly Ajax: The Definitive Guide (Ajax 权威指南)

Beginning Javascript with DOM Scripting and Ajax 从入门到精通

Accelerated DOM Scripting with Ajax, APIs, and Libraries

Ajax Patterns and Best Practices

Head Rush Ajax

O'Reilly Ajax Hacks

O'Reilly Adding Ajax

Practical JavaScript DOM Scripting and Ajax Projects

Ajax - A New Approach to Web Applications

SEO: Search Engine Optimization Bible

AJAX 基础教程 AJAX Essential Training 视频教程系列

大家网[www.topsage.com]

超强 HTML 和 xhtml，CSS 精品学习资料下载汇总

HTML/CSS 开发工具：

最强 CSS 设计工具 NewsGator TopStyle Pro v4.0 最新版＋序列号

HttpWatch v6.0.14 Pro (附件授权文件)

文本转 HTML 工具：Easy Text To HTML Converter

Beginning Web Programming with HTML, XHTML and CSS 第二版

专业网页设计工具 Blumentals WeBuilder 2008 v9.2.0.100

快速 CSS 编写工具 Blumentals Rapid CSS 2008

专业 HTML 编写工具 Blumentals HTMLPad 2008 Pro

UI 原型设计工具 Axure RP Pro v5.6.0.2158/Win/含注册机

UI 原型设计工具 Axure RP Pro v6.0.0.2890/Win/含注册机

W3C 官方手册：

CSS 完全参考手册 3.0

xHTML 完全参考手册 5 合 1 W3C 官方权威手册

HTML 4.01 规范 - W3C 官方 HTML 权威指南

HTML 4.0 参考手册 CHM

W3C HTML 3.2 规范

CSS 2 权威 W3C 官方参考手册 CHM

经典资源推荐：

Web 编程入门经典：HTML、XHTML 和 CSS (第 2 版)

HTML 与 CSS 入门经典（第 7 版）

Web 编程入门经典——HTML、XHTML 和 CSS（第 2 版）

网页制作基础 视频教程 swf 格式全 33 讲

CSS 彻底研究 swf 视频教程 全 23 讲下载

Eric Meyer 谈 CSS (卷 1)

Eric Meyer 谈 CSS (卷 2)

高性能网站建设指南——前端工程师技能精髓

无懈可击的 Web 设计

完美 HTML 设计 - 使用 CSS 不用 Table (第二版)

XHTML 实例精解

XHTML 技术内幕

HTML, XHTML, and CSS Bible, 5th Edition

精通 XHTML 程序设计高级编程

XHTML 视频教程：Lynda.Com Learning XHTML - Lynda.com 视频教程系列

彻底设计研究 CSS

CSS 时尚编程百例

HTML 简明教程 中文 PDF 版

Designing with Web Standards 网站重构 英文+中文版

CSS 禅意花园 (高级 CSS 开发)

HTML & XHTML 权威指南(英文+中文版)

Don't Make Me Think!

Web 信息架构——设计大型网站(第 3 版)

Apress Pro CSS and HTML Design Patterns

O'Reilly CSS: The Definitive Guide 第三版(CSS 权威指南)

Ultimate HTML Reference

HTML & XHTML 权威编程 第五版

The CSS Anthology (第三版)

The CSS Anthology (第二版)

Professional CSS: Cascading Style Sheets for Web Design 第二版

Head First HTML with CSS & XHTML

Dreamweaver 8 ASP 动态网站建设精粹【视频教程下载】

HTML5 & CSS3：

Sams Teach Yourself HTML5 in 10 Minutes (5th Edition)

HTML5 and CSS3: Develop with Tomorrow's Standards Today

O'Reilly HTML5: Up and Running

Stunning CSS3: A project-based guide to the latest in CSS

The Definitive Guide to HTML5 Video

Canvas Pocket Reference: Scripted Graphics for HTML5

CSS3 先睹为快视频教程

HTML5 先睹为快 视频教程

HTML5 教程： 先睹为快

HTML5 结构语法和语义视频教程

Apress.Pro.HTML5.Programming.Sep.2010

css3_for_web_designers

Dive Into HTML5

HTML5 高级程序设计

HTML5.and.CSS3： Develop with Tomorrow's Standards Today

HTML5 Step by Step

A Book Apart CSS3 For Web Designers

HTML5 与 CSS3 权威指南第 1 章~第 3 章

The Book of CSS3: A Developer’s Guide to the Future of Web Design

其他 HTML/CSS 参考资料：

Sams Teach Yourself HTML 4 in 10 Minutes

Wrox Web Standards Programmer's Reference

Accessible XHTML and CSS Web Sites: Problem - Design - Solution

Wrox CSS Instant Results

CSS Mastery: Advanced Web Standards Solutions

Creating Cool Web Sites with HTML, XHTML, and CSS

Beginning HTML with CSS and XHTML: Modern Guide and Reference

O'Reilly CSS The Missing Manual

Wiley Creating Web Sites Bible

Website Optimization: Speed, Search Engine & Conversion Rate Secrets

Accessible XHTML and CSS Web Sites

Speed Up Your Site: Web Site Optimization

Spring into HTML and CSS

O'Reilly Designing Web Navigation

Beginning CSS: Cascading Style Sheets for Web Design

The Ultimate CSS Reference

The Essential Guide to CSS and HTML Web Design

Beginning CSS Web Development 从入门到精通

Apress Pro JavaScript Design Patterns

The Art & Science of JavaScript

Teach Yourself HTML in 10 Minutes 第四版

HTML,XHTML,and CSS - Visual Quickstart Guide 第六版

HTML & XHTML - The Complete Reference 第四版

Assembly Language Step by Step

How to Do Everything with Web 2.0 Mashups

Build Your Own Database Driven Website

XHTML Moving toward XML

Spatial Data on the Web:Modeling and Management

How to Do Everything With HTML

Beginning HTML with CSS and XHTML

O'Reilly CSS Cookbook

Adobe Dreamweaver CS3 Bible

CSS Web Development 从入门到精通

Pro CSS Techniques

The Best Practice Guide to xHTML and CSS

Build your Own WebSite - The Right Way Using HTML and CSS

Mastering Integrated HTML and CSS

Dreamweaver CS5 创建管理 CSS 视频教程 Lynda.com Dreamweaver CS5: Managing CSS

网页开发工具：JetBrains WebStorm v2.0.1

深入浅出 HTML（英文版）

Transcending.CSS（英文版）

Pro CSS for High Traffic Websites

Beginning iPhone and iPad web Apps

CSS 精粹(第 3 版)

Web 导航设计

Web 表单设计:点石成金的艺术

Beginning iPad and iPhone Apps with HTML5 CSS3 and JAVAScript

HTML Manual of Style 4th

博客园精华集 Web 标准之道

编写高质量代码-Web 前端开发修炼之道（完整版）

变幻之美——Div+CSS 网页布局揭秘(案例实战篇)

27 款经典的 CSS 框架

分享到新浪微博/QQ 空间/开心网/人人网/豆瓣网/QQ 书签/百度搜藏/美味书签 代码

国外在线调色工具、在线 CSS 工具、CSS 设计展示网站汇总

网页设计师一定要知道的 30 多个 CSS 和网站设计画廊网站

韩国网页设计素材包免费下载

网页常用素材打包下载【800 张网页背景素材、1000 个网页小图标、1000 张网页常用图片】

CSS 时尚编程百例

100 款精美的 css+html 模板

网页配色手册

教程地址汇总

31 个用来测试你网站各项性能的在线工具

近 6000 个 Web 2.0 网站和模板欣赏

超酷韩国网页模板 全部 PDF 格式原版文件 (19 套)

推荐几个国外比较好的 CSS 设计模板网站

25 种方式给你的网站添加优质内容

网页设计师必备资源网站

素材实例下载：

网页设计必备的 10 套漂亮图标集

20 款绝不能放过的 HTML5 应用程序示例

HTML 颜色代码表

WEB 素材..CSS、FLASH...等素材代码

前　言

设计人员是典型的创造型人才，偏爱通过右脑思考，而注重技术细节的程序员则偏爱于用左脑思考。

但当面对Web设计的挑战时，设计人员则会运用称为CSS(Cascading Style Sheet，层叠样式表)的设计技术。

CSS是由Worldwide Web Consortium (W3C)制定的一个Web标记标准集，该标准集用于定义Web页面中的一致样式，并把该模板应用于多个页面。CSS本质上是一种技术，这种技术在很大程度上必须通过手工编写以创建一些强制性规则。这样大多数设计人员要完成自己的工作就必须编写大量的代码。

我们不需要只会手写PostScript的设计人员，而是需要能利用Adobe Illustrator提供的可视化编程环境(代码隐藏在背景中)来编写PostScript文件的设计人员。

当WYSIWYG(What you see is what you get，所见即所得)Web页面编辑人员的可视化编程经验越来越丰富时，这些应用软件不再是真正的专业CSS设计工具，如Illustrator中的PostScriptv那样。

CSS还存在一个与浏览器兼容的问题，但PostScript不存在这样的问题。浏览器开发商在其浏览器中没有及时支持这种技术。即使对CSS的支持越来越好(特别是在IE 7 for Windows中对CSS的良好支持)，但当设计人员试图在老的或过时的浏览器上支持他们的设计时仍会遇到问题。这意味着有必要深入研究CSS，并编写一些hack(在本书指一些修复浏览器错误的代码)和一些迂回解决方案。

也就意味着应该花更多的时间来编写和修改代码，而不应该把过多时间花在WYSIYG工具的使用上。

即使了解CSS的基本元素(属性、取值、选择符等)，要充分利用这项技术也是很困难的，甚至会使人感到沮丧。CSS会妨碍我们中的大部分人，不论是对W3C规范有很好了解的人，还是那些怀着畏惧心理浏览这些规范的人。

从正面角度看，CSS确实是一种工具。

一旦设计人员掌握了该技术的基本元素，了解了其目的，并获得了该技术的足够经验，几乎任何一个画在餐巾纸上或用Adobe Photoshop完成的设计思想都可以被实现。

要达到这一点，您必须阅读本书。

对于大规模的、设计良好的多页面网站，强调以集成方式运用CSS设计出基于标准的CSS，而本书就是当今市场上倡导这种思想的为数不多的几本书之一。

针对网站开发中遇到的设计问题，本书通过网站开发的最佳实践经验和大量的网站实例，提供了独特的、CSS驱动的解决方案。

本书读者对象

本书适用于对CSS有所了解的中级和高级设计人员。如果您对如何以专业水平高效地开发CSS驱动的设计不是很清楚，学习本书将收益匪浅。特别地，本书对下列读者非常有用。

- 刚刚接触CSS并有一定HTML编程经验的中级用户——对任何一个专业的Web开发人员而言，掌握CSS都是很容易的，但要用CSS创建专业水平的Web站点则需要对CSS有非常深入的了解。
- 专业设计人员——学习过CSS的专业Web开发人员(没有20世纪90年代传统设计经验知识)并希望了解使用这种技术的最佳实例的专业Web开发人员。

本书主要内容

本书每一章都集中介绍一位设计人员以及他所设计的Web站点。每一章都提供一些容易接受的CSS技巧演示和开发网站所用到的技术。另外，设计人员通过介绍开发过程中的不同做法向读者提供了一些更深邃的见解。

下面简单介绍一下本书主要内容和合著者贡献的深刻见解。

- 第1章：“有关XHTML和CSS的最佳实例”——Ethan Marcotte，WSP(Web Standards Project)的权威委员、基于标准的Web设计运动的组织领导者。他将在该章与大家分享一些使用含CSS的XHTML(eXtended HTML，XHTML)的深刻见解。
- 第2章：“Google的blogger.com：翻转器和设计思想”——Dunstan Orchard，也是WSP的一位委员，他对blogger.com (一个Google Web站点) 外观的实现内幕进行了深入研究。Douglas Bowman(资深设计人员，极力宣扬对一些Web站点进行重新设计，并取得了巨大成功，这把他推到设计与标准相兼容的Web的前沿)是该项目的

负责人之一，Orchard对他进行了访谈，给我们提供了一些极其有价值的见解。

- 第3章：“经典的美国职业高尔夫联盟锦标赛网站”——Todd Dominey，TSI(Turner Sports Interactive)的高级互动设计人员，他为大多数职业高尔夫联盟(Professional Golf Association，PGA)巡回赛设计和开发了相应的Web站点，包括PGA冠军杯和莱德杯(Ryder Cup)。该章将讲述如何根据世界各地球迷需求进行网站设计的细节，Dominey将为读者讲述独家观点。该章着重介绍的关键技术包括阴影、下拉菜单和如何把Flash内容嵌入到一个Web站点。
- 第4章：“佛罗里达大学主页UFL.edu”——Mark Trammell，负责指导美国一所著名大学的Web互动网站。他介绍了佛罗里达大学是如何从开发Web站点的过程中使学生和机构受益的。该章着重介绍的关键问题包括处理与浏览器兼容的问题和导航功能结构。
- 第5章：“Stuff and Nonsense：CSS切换策略”——除与Andy Clarke(富有创造性的Stuff and Nonsense项目组负责人)的访谈外，Ethan Marcotte还对如何改进Web站点的可访问性进行了探讨，以进一步保证了所有用户能以统一的方式访问Web站点。在该章，Marcotte仔细研究了CSS切换和如何克服令人讨厌的浏览器兼容问题。在Stuff and Nonsense站点所体现出来的创新是运用这些技术的优秀范例。
- 第6章：“CindyLi.com的风险投资：博客修改”——Cindy Li介绍了她是如何通过她的设想和CSS编码定制自己的Web站点的。
- 第7章：“辛辛那提的AIGA：HTML email模板”——Christopher Schmitt先简单介绍了创建email模板的过程，接着介绍了如何创建一个基本的HTML布局表，并说明了CSS所起的关键作用。
- 第8章：“专业CSS图书网站：透明PNG图像的使用”——Christopher Schmitt介绍了如何用PNG图像创建该图书网站，以及如何解决IE 6对PNG图像的alpha透明缺乏良好支持的问题。
- 第9章：“构建CSS布局”——Christopher Schmitt介绍了网格和布局在设计中的重要性，Ethan Marcotte探讨了如何创建一个三栏布局的稳定架构。

另外，本书附录还提供了有关HTML 4.01元素、HTML到XHTML的转换规则、CSS 2.1属性的参考资料以及一个排除常见问题的帮助指南。

源代码

在读者学习本书中的示例时，可以手工输入所有的代码，也可以使用本书附带的源代码文件。本书使用的所有源代码都可以从本书合作站点http://www.wrox.com/(或http://www.tupwk.com.cn/downpage)上下载。登录到站点http://www.wrox.com/，使用Search工具或使用书名列表就可以找到本书。接着单击本书详情页面上的Download Code链接，就可以获得所有的源代码。

注释：

由于许多图书的标题都很类似，所以按ISBN搜索是最简单的，本书英文版的ISBN是978-0-470-17708-2。

在下载了代码后，只需用自己喜欢的解压缩软件对它进行解压缩即可。另外，也可以进入http://www.wrox.com/dynamic/books/download.aspx上的Wrox代码下载主页，查看本书和其他Wrox图书的所有代码。

勘误表

尽管我们已经尽了各种努力来保证文章或代码中不出现错误，但错误总是难免的，如果您在本书中找到了错误(例如拼写错误或代码错误)，请告诉我们，我们将非常感激。通过勘误表，可以让其他读者避免受挫，当然，这还有助于提供更高质量的信息。

请给wkservice@vip.163.com发电子邮件，我们就会检查您的反馈信息，如果是正确的，我们将在本书的后续版本中采用。

要在网站上找到本书英文版的勘误表，可以登录http://www.wrox.com，通过Search工具或书名列表查找本书，然后在本书的详情页面上，单击Book Errata链接。在这个页面上可以查看到Wrox编辑已提交的所有勘误项。在完整的图书列表中还包括每本书的勘误表，网址是http://www.wrox.com/misc-pages/booklist.shtml。

p2p.wrox.com

要与作者和同行进行讨论，请加入p2p.wrox.com上的P2P论坛。这个论坛是一个基于Web的系统，便于您发布与Wrox图书相关的消息和相关技术，与其他读者和技术用户交流

心得。该论坛提供了订阅功能，当论坛上有新的消息时，它可以给您传送感兴趣的论题。Wrox作者、编辑、其他业界专家和读者都会到这个论坛上来探讨问题。

在http://p2p.wrox.com上，有许多不同的论坛，它们不仅有助于阅读本书，还有助于读者开发自己的应用程序。要加入论坛，可以遵循下面的步骤：

(1) 进入p2p.wrox.com，单击Register链接。

(2) 阅读使用协议，并单击Agree按扭。

(3) 填写加入该论坛所需要的信息和自己希望提供的其他信息，单击Submit按扭。

(4) 您会收到一封电子邮件，其中的信息描述了如何验证账户，完成加入过程。

注释：

不加入P2P也可以阅读论坛上的消息，但要发布自己的消息，就必须加入该论坛。

加入论坛后，就可以发布新消息，回复其他用户发布的消息。可以随时在Web上阅读消息。如果要让该网站给自己发送特定论坛中的消息，可以单击论坛列表中该论坛名旁边的Subscribe to this Forum图标。

关于使用Wrox P2P的更多信息，可阅读P2P FAQ，了解论坛软件的工作情况以及P2P和Wrox图书的许多常见问题。要阅读FAQ，可以在任意P2P页面上单击FAQ链接。

目　录

第1章

有关XHTML和CSS的最佳实例

在早期，Web并不是最吸引人的事物。由核物理学家创建和使用的超文本只不过是在开放的分布式网络中共享文档的一种手段，而且这些文档的主要内容是文本。不必讳言，高品质的设计并不是早期Web开发者所优先考虑的目标。事实上，常用的HTML表格元素(也是经常被滥用的，在后面将介绍)是为一个目的而创建的：显示表格数据。

20世纪90年代后期是Web设计的全盛时期。在该时期，HTML中的“L(Language)”常常被忽视。许多专业人士觉得构建Web页面的代码在本质上并不是一种语言，因此不受真正编程语言的规则和约束的限制。此外，用户并不会为兼容的、可读性好的或防过时的代码支付费用。事实上，许多站点是根据“向后兼容”的要求开发的。这可能用词不当，因为曾经有这样的站点，只支持IE 4.0或以上版本的浏览器。

至少可以说，那个时期的浏览器性能是不稳定的。由于对浏览器W3C(http://www.w3.org)制定的规范支持很差，可能同一个页面在浏览器A和浏览器B中的显示结果不一样。因此，当许多程序员朦朦胧胧知道W3C制定的“标准”时，他们必须支持的浏览器还很少能容忍与标准兼容的标记。在这种意义上，把Web内容和形式进行分离的要求很迫切。对于HTML中与浏览器相关的专有标记，程序员故意使它无效以保证在目标浏览器中“看起来是好的”。所以一切暂时相安无事。程序员抱怨需求规格说明不准确、工期很紧，而且他们可能不是按钟点计酬，总之，他们有很多借口。

当然，对于以前学习过如何把表格的单元、间距和边框设置为零的设计人员，可以构建基于网格的复杂布局，使他们的站点在美学感染力方面到达一个新的高度。不可否认，由于在这个时期的浏览器对CSS的支持很差，程序员除了删除目前不用的表现代码来减小页面大小外别无选择。这导致Web自身的标记太多，页面大小以千字节计。对这样的页面而言，维护困难、重新设计的成本非常高，而且还会浪费用户的带宽。

令人欣慰的是，这样的时代过去了。扩展超文本标记语言(eXtensible Hypertext Markup Language，XHTML)和CSS这两个标准技术将消除页面中的混乱，使页面的构建工作变得越来越轻松。页面占用空间也越来越小、可读性越来越好、越来越易于维护。当然，只有在完全掌握这两个工具后才能充分发挥它们的作用。本章对引入XHTML和CSS的必要性进行了分析。为了把它们灵活运用到设计中，还介绍了几个实用策略。

1.1 把结构和表现标记硬挤在一起

现在屏住呼吸，诚实地问一下自己：熟悉下面一行HTML代码吗？

```
<body marginwidth="0" marginheight="0" leftmargin="0" topmargin="0" >
```

在Web设计的全盛时期，这是把页面内容放入整个浏览器窗口的方法。没有这四个属性，设计的页面将被10个像素左右的边界所环绕——是的，类似这样的事有点太挑剔了。

强调“看起来是对的”在早期的Web设计中非常普遍，这个方案说明了这种普及程度。尽管HTML源于一种结构良好的语言，但页面已发展为一种“标签汤(tag soup)”——把结构和表现标记混在一起的页面(就像一种味道不是很好的菜炖牛肉)。由于当时的浏览器不支持CSS或支持不够，因此程序员只有依靠透明图形、字体元素和多重嵌套表来控制站点的设计。body元素包含了大量属性，很能说明这种在标记中结构和样式不匹配的问题。虽然body元素本身主要用于结构设计(它包含Web页面的内容)，但在其开始标签中的少量属性只是使结构看起来像那么回事。

必须承认，小小的body元素似乎没有如此神奇——真的值得我们关注标记的这一行吗？为了讲述标记中的问题，我们来看看哈佛大学主页(http://www.harvard.edu)这样的具体范例。该网站的设计(如图1-1所示)很好地反映了哈佛大学已树立的良好品牌：保守、土色调的色条着重强调了独特的哈佛红，同时中央的两列布局增加了内容的可读性。据大家反映，这是一个成功的站点设计——承载的信息非常丰富。

图1-1　哈佛大学主页

显然这是一个直观有效的设计。但是，如果把HTML中所有表格元素的边框都显示出来，表现的效果就差多了(如图1-2所示)。

说明：

现在有很多浏览器实用工具。安装这些工具后可影响页面的显示，正如这里的例子一样。对Molliza浏览器，Web Developer Toolbar (http://chrispederick.com/work/firefox/webdeveloper)就是这样的工具，而且是一个非常优秀的工具。它是CSS工具集中极其有价值的一部分。设计人员可用它激活不同页面元素的边框、快速编辑页面的CSS、很容易地获得各种在线代码验证软件。对于Mac平台的Web开发人员，可在下列网址中找到相应插件：http://hicksdesign.co.uk/journal/web-development-with-safari。

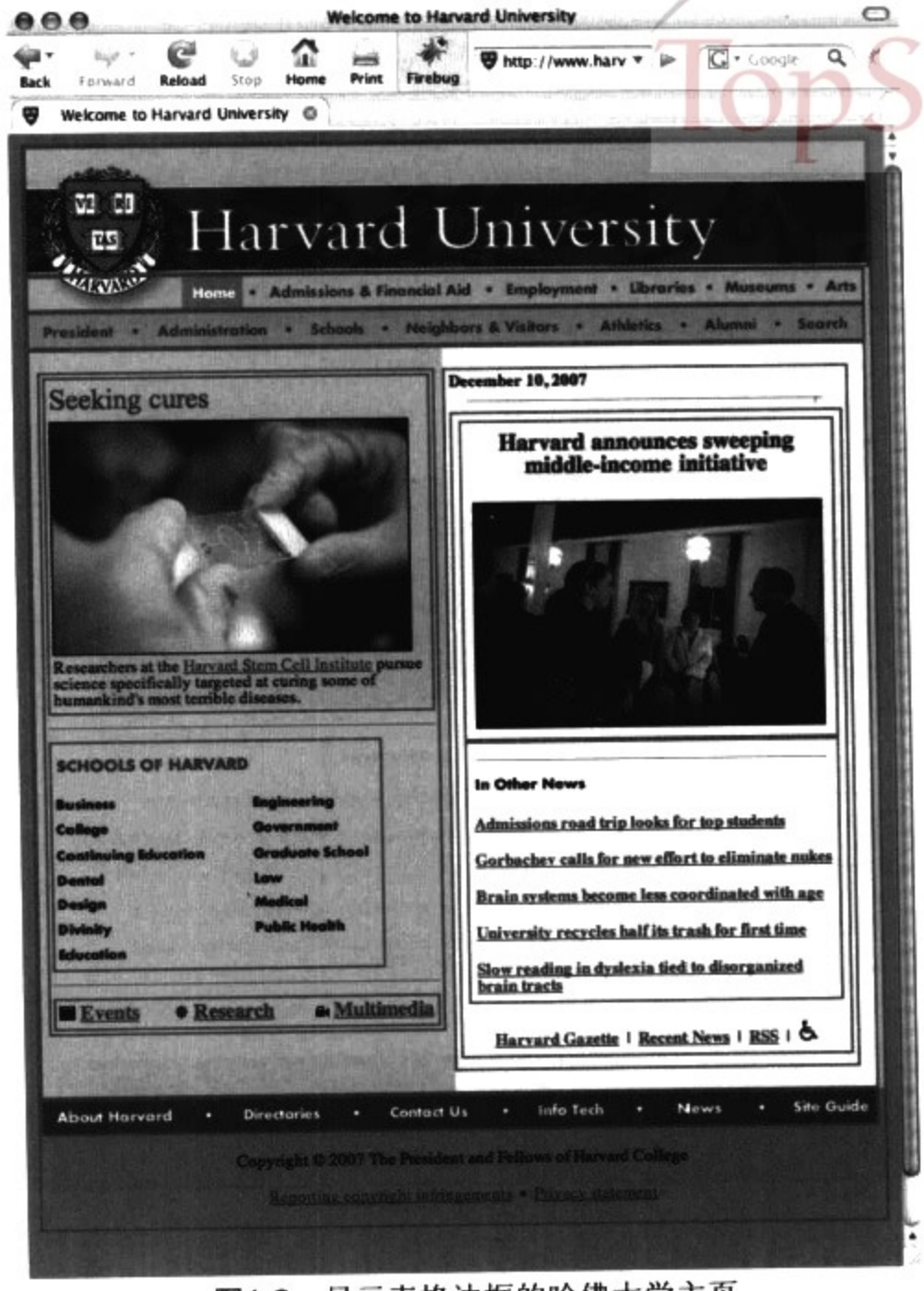


图1-2 显示表格边框的哈佛大学主页

变化很大，不是吗？在如此简单的布局中就有如此多的标记：表格嵌套深度为五层；logo图标被分解到多个文件并散布在多个表格行中。即使查看一下简单的导航条，也有点使人晕眩。

```
    <table bgcolor="#cdd397" border="0" cellpadding="0" cellspacing="0"
width="650">
    <tbody><tr>
    <td valign="top"><img src="images/shield3.gif" alt="Harvard University
shield" border="0" height="25" width="117"></td>
```

```
    <td valign="top"><a href="http://www.harvard.edu/"><img src="images/
home2.gif" alt="Home" name="nav01" border="0" height="25" width="47"></a></
td>
    <td><img src="images/nav_bullet.gif" border="0" height="25"
width="14"></td>
    <td><a href="http://www.harvard.edu/admissions/"
onmouseover="imgOn('nav02')" ;=""onmouseout="navOff('nav02')"><im
g src="images/admissions.gif" alt="Admissions" name="nav02" border="0"
height="25" width="166"></a></td>
    <td><img src="images/nav_bullet.gif" border="0" height="25"
width="14"></td>
    <td><a href="http://atwork.harvard.edu/" onmouseover="imgOn('nav03')"
;=""onmouseout="navOff('nav03')"><img src="images/employment.gif"
alt="Employment" name="nav03" border="0" height="25" width="80"></a></td>
    <td><img src="images/nav_bullet.gif" border="0" height="25"
width="14"></td>
    <td><a href="http://lib.harvard.edu/" onmouseover="imgOn('nav04')" ;=""
onmouseout="navOff('nav04')"><img src="images/libraries.gif" alt="Libraries"
name="nav04" border="0" height="25" width="59"></a></td>
    <td><img src="images/nav_bullet.gif" border="0" height="25"
width="14"></td>
    <td><a href="http://www.harvard.edu/museums/" onmouseover= "imgOn
('nav05')" ;="" onmouseout="navOff('nav05')"><img src="images/museums.gif"
alt="Museums" name="nav05" border="0" height="25" width="64"></a></td>
    <td><img src="images/nav_bullet.gif" border="0" height="25"
width="14"></td>
    <td><a href="http://www.harvard.edu/arts/" onmouseover="imgOn('nav06')"
;="" onmouseout="navOff('nav06')"><img src="images/arts.gif" alt="Arts"
name="nav06" border="0" height="25" width="33"></a></td>
    </tr>
    </tbody></table>
```

首先设置表格第一个导航条的背景色(#cdd397，不饱和浅绿色)并把表格边框设置为0，同时把每个单元格的边距和间距也设置为0。一旦完成上述设置就会形成一个不可见的网格，其中放置的图形可精确到像素。其余的每个表格单元都包含nav_bullet.gif，每个图形都与一个导航项对应。其余单元格包含导航图片本身，每个导航图片都被一个锚包围，该锚包含Onmouseover和Onmouseout两个属性，用于控制图形的交替显示效果。

注意，这仅仅是其中一个导航条的标记。该页面其余部分遵循同样的布局模式：把一个表格的默认属性设置为0；在其中放置内容、图形和附加标记；根据需要重复上述步骤。只有我们认为到达了一个表格的末尾，另一个表格才会出现，这使我们想起搜寻“看

起来正确”的圣杯——在所有目标浏览器中都有真正一致、预防出错的显示——需要付出多大的努力。

当然，圣杯是一个小锡杯。到目前为止，设计人员只关心站点在图形化桌面型浏览器上的可视化显示。但还应该考虑其他设备和用户。如果在不能很好渲染点阵图形的环境中浏览哈佛大学的主页，会有什么现象发生？

在纯文本的浏览器上浏览该站点的结果如图1-3所示。没有颜色和标题行的帮助，在该环境中进行导航确实比在站点设计语境进行导航要困难得多。如果对视力正常的用户都如此困难，那么设想一下盲人用户会如何面对这样一个页面。

例如，该主页的大量图形失去了alt属性，这是一个很重要的可访问性需求，也是有效XHTML所需要的。只有创建有效的标记，才更容易把CSS应用到Web文档。

如果盲人用户用屏幕阅读器朗读Web页面的内容，导航菜单听起来是：“Home链接nav下划线bullet点gif，Admission链接nav下划线bullet点gif，Employment链接nav下划线bullet点gif，Libraries链接”，等等。

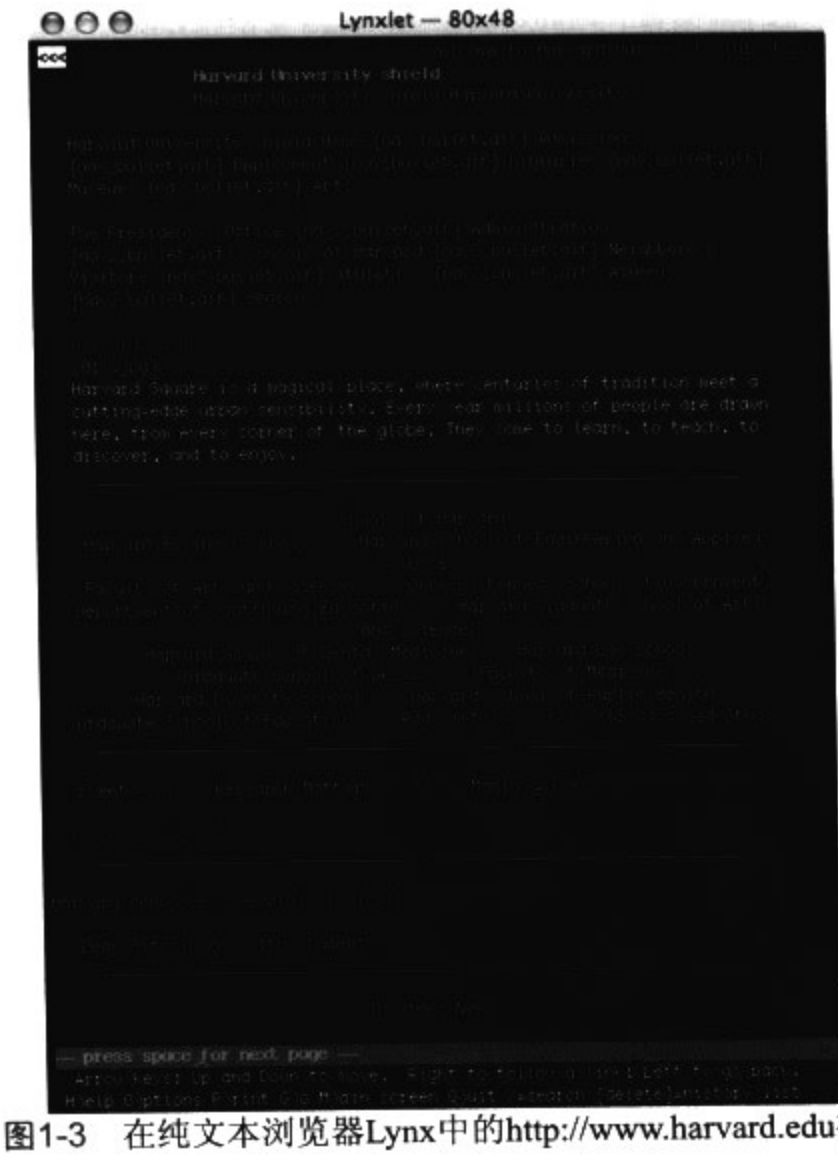


图1-3　在纯文本浏览器Lynx中的http://www.harvard.edu视图

首先，这不能归罪于哈佛主页；在过去的岁月里，大量页面也正是按照这种策略构建的。事实上，这正反映了Web的现实情况，直到最近所有东西才被引进。正因为对CSS的支持不够，基于表格的布局是必然的结果。设计人员把精力放在了如何保证在所有图形浏览器中都有非常好的显示效果，为此由于标记繁多而带来带宽和站点设计可读性等方面的代价。我们如何解决这个问题？

目前的浏览器变得越来越智能，这也是大家所希望的。随着普遍对层叠样式表越来越

好的支持，不必再通过添加大量浪费带宽的标记来实现站点设计。取而代之的是把用于表现的代码从标记中提取出来，放到层叠样式表中。

对结构与样式分离的承诺是根据标准进行Web设计的核心。这使得用CSS构建页面布局成为最主流的观点之一。Web页面的内容和表现形式的分离，不仅使页面越来越精巧，而且更易于维护。

1.2　学习并热衷于使用标记

让我们再来回顾一下body元素：

```
<body marginwidth="0" marginheight="0" leftmargin="0" topmargin="0">
```

值得注意的是，这些属性在任何HTML规范(http://www.w3.org/MarkUp)中都找不到。只有Netscape使用属性marginwidth 和 marginheight，这两个属性也只在Netscape浏览器中发挥作用。另一方面，属性leftmargin 和 topmargin与marginwidth 和 marginheight有相同的边界对齐效果，不过前者只在IE中才能发挥作用。无论有效与否，都不能阻止程序员把这个私有标记放到站点中。他们正处理的浏览器所实现的HTML是非标准的(常常是矛盾的)，他们还不得不面带微笑地这样做。

又由于他们把同样无效、私有的HTML提供给所有访问该页面的浏览器，HTML的这一行说明了浏览器是如何容忍有缺陷的标记的。这就是设计工作所不得不考虑的。如果因为疏忽而没有包含结束标签(如</tr>或)，或为了解决布局方面的缺陷而在标记中引入私有元素(如前面提到的body元素)，Web浏览器有适当的恢复策略。在解析和呈现这样的页面时，尽管发现有不完美的代码，浏览器也不得不修复这些标记。

但真正令人头疼的是，每个浏览器都有自己一套解释无效代码的内部逻辑，这在跨浏览器测试中将出现不可预测的结果。一个浏览器或许可以修复因丢失了尖括号带来的问题，而另一个浏览器则可能因此停止显示整个布局。这种不一致的呈现策略必然带来：无效的代码意味着要花更多的编码和测试时间，以保证在所有目标浏览器中都能被正确显示。这样设计人员就不能集中精力来改进站点的设计和内容，而不得不花大量时间来关心不完整的标记。事实确实如此。早期靠畸形、私有或其他无效标记使设计在老式的浏览器上能有一致的显示。但是利用标记修复，设计人员在站点的每个页面建立一个依赖关

系。他们在文档中加入代码来获取指定浏览器的特征。当目前的浏览器完成最终升级和下一代浏览器出现时，每一个标记修复都将成为潜在的地雷。更新的浏览器能容忍这些塞满非标准标记的代码吗？或者，下一版本的IE会隐藏这样的错误：当它遇到body元素中的marginheight属性时会拒绝呈现吗？

是的，这也是需要考虑的。但显然无效标记造成的负担是长期的、不能忽略的。与其对未来的一年或八年内标记修复能否一直被支持而焦虑，不如把注意力放在HTML中的“L”上。

1.2.1 XHTML：新热点

W3C把XHTML描述为：“把XML的严密引入到Web页面”(http://www.w3.org/MarkUp/#xhtml1)。简而言之，XHTML为站点所有者提供了一条从HTML升级到XML(XML具有更严格的文档语法)的清晰路线。比较下列HTML片段及其等价的XHTML片段。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
 <title>My sample page</title>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
<ul>
 <li>Here's a sample list item,
 <li>And yet another.
</ul>
<p>I like big blocks,<br>and I cannot lie.
<p>You other coders can't deny.
</body>
</html>
```

再来看看等价的XHTML片段。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
```

```
    <title>My sample page</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  </head>
  <body>
    <ul>
      <li>Here's a sample list item,</li>
      <li>And yet another.</li>
    </ul>
    <p>I like big blocks,<br />and I cannot lie.</p>
    <p>You other coders can't deny.</p>
  </body>
</html>
```

如果没发现什么变化，不用感到沮丧——它们其实非常类似，并没有什么差别。事实上，这两个页面在任何浏览器上显示的效果完全一样。终端用户并不能分辨出这两种语言之间的差别。XHTML在语法上与HTML的类似之处远远多于它们之间的差别，实际上这种差别非常小。

1. DOCTYPE声明

上述XHTML样例页面的第一行是DOCTYPE声明。在DOCTYPE元素中把浏览器指向如下的URL：

```
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd
```

根据所用浏览器和计算机对.dtd文件扩展名的处理方式，可能会提示您下载或在浏览器窗口中以纯文本方式显示该文件。不论是在浏览器还是文本编辑器中浏览该文件，都会呈现一个长长的、复杂的、被称为文档类型定义(Document Type Definition，DTD)的文本文档。DTD描述了标记应遵循的语言规则。在标记一开始声明所用的DTD，通知“用户代理”该页面所用的语言。“用户代理”是一个广义术语，指可访问该页面的所有设备和程序，不仅限于图形化的桌面浏览器。打印机、移动电话和屏幕朗读器都是用户代理的实例。事先知道该页面后面所用的标记语言，对这些用户代理都非常有好处。

联机有效性验证工具也是用户代理的实例，与其他客户代理相比，这些工具将从起始的DOCTYPE声明中获得更大益处。这些验证工具可对代码遵循该DTD的有效程度进行评估——不论是否有效。

2. 保持标记拥有良好架构

“良好架构”对旧规则来讲是一个非常重要的新术语。良好架构仅仅意味着所用元素必须被正确嵌套。请看下面的例子：

```
<p>Here's <em>my <strong>opening</em></strong> paragraph!</p>
  <title>My sample page</title>
```

在这里，首先打开em，然后打开strong。然而标记遵循先打开后关闭的原则。由于em在strong之前打开，因此它必须在strong结束标签之后关闭。如果要把这段标记修改为良好架构的，需要对元素的嵌套顺序做一点小小改变：

```
<p>Here's <em>my <strong>opening</strong></em> paragraph!</p>
  <title>My sample page</title>
```

毫无疑问，您已经注意到，正确嵌套是一个已有的概念。书写不正确的嵌套标记永远都是无效的，这种标记在今天的许多页面构建中仍非常普遍。结果，浏览器越来越能忍受喂给它的标签汤。任何一个给定的浏览器都有各自的策略对嵌套不正确的标记进行修复，所以呈现页面时经常产生不同的结果。XHTML是一种显式要求结构正确的语言。通过对标记的良好架构要求，使文档语法更严格，这样在代码中就可消除结构的不一致性。

当然，良好架构并不等于有效，认识到这一点很重要。请看下面的例子：

```
<a id="content">
 <em>
  <div class="item">
   <p>I'm not exactly sure what this means.</p>
   <p>...but at least it's well-formed.</p>
  </div>
 </em>
</a>
```

这段代码是正确嵌套的，但不是有效的。HTML有块级元素(如div、p、table等)和内联元素(如a、em、strong等)之分。内联元素永远不能包含块级元素，这条规则使得前面的标记是无效的。虽然浏览器能正确解读这段代码，但在应用CSS时几乎肯定会导致显示错误。不同的浏览器，应用于锚元素的样式可能会(也可能不会)把包含在div中的文本向下层叠。当鼠标停留在链接之上时，如果所有内容都带下划线或发生其他显著变化，您一定会感到惊奇和不快。

这也是说明有效性验证重要性的另一实例。在文档开始时进行DOCTYPE声明和经常

进行有效性验证，就能把布局方面的缺陷扼杀在萌芽状态。这样就可以大大减少调试的时间，从而把更多精力放在站点设计上。

3. 关闭每一个元素

在HTML中打开一个元素时，并不总是要求提供相应的结束元素。事实上，HTML 4.01规范按结束元素是可选的(如段落元素: http://www.w3.org/TR/REC-html40/struct/text.html#h-9.3.1)、必需的(如em、strong这样的段元素：www.w3.org/TR/REC-html40/struct/text.html#h-9.2.1)和在某些情况下是完全禁止的(http://www.w3.org/TR/REC-html40/struct/text.html#h-9.3.2)对元素进行了区分。

令人欣慰的是，在XHTML中没有这种不确定性，这主要归功于XHTML坚决主张标记必须是良好架构的。如果打开一个元素，就必须有结束元素——就是这样简单。在HTML4中，下列代码是合法的标记：

```
 <ul>
  <li>Here's a sample list item,
  <li>And yet another.
</ul>
<p>I like big blocks,<br>and I cannot lie.
<p>You other coders can't deny.
```

然而在XHTML中就稍微有些不同(变化之处由粗体标出)：

```
<ul>
  <li>Here's a sample list item,</li>
  <li>And yet another.</li>
</ul>
<p>I like big blocks,<br />and I cannot lie.</p>
<p>You other coders can't deny.</p>
```

由于是在XHTML中，就必须使列表项(
)和段落( )是闭合的。在开始一个新元素之前，需要用相应的
和 关闭每一个元素。眼尖的读者可能会感到奇怪，为什么在br后面加一斜杠(/)？

诸如br、img、meta和hr这样的元素被认为是空元素，因为它们不包含任何文本内容，这与大多数HTML规范中的元素不同，如p、li和td等。但是，在传统上空元素没有结束元素，而XHTML为了保持文档的良好架构就不能有例外。所以给空元素加上“/>”来就可有效地关闭它。结构一致性对新标记是一种很强的要求，而XHTML确实满足这种一致性要求。

说明：

注意到前一范例中<br和/>之间的空格吗？这个空格确保老的浏览器(在XHTML规范之前开发的)仍能访问到您的内容。

4. 把元素和属性设置为小写

HTML规范对页面的标记元素的大小写从没做强制要求。因此，开发人员习惯于以任何形式书写元素和属性：

```
<P CLASS=Warning>Alert!</P>
  <title>My sample page</title>
```

在XHTML中，所有元素和属性必须是小写的。这是因为XML是区分大小写的。例如，<body>、<Body>和<BODY>被认为是三个不同的元素。基于这个原因，XHTML规范的制定者把小写制定为标准：

```
<p class="Warning">Alert!</p>
  <title>My sample page</title>
```

您可能注意到class属性的值Warning没有进行大小写转换。这在XHTML中是完全可以接受的，因为属性值可以是大小写混合的(例如，一个链接的href，它指向区分大小写的服务器)。但是属性值必须用引号引起来。

5. 必须为每个属性指定一个值

另外，以前的HTML中存在不需要属性值的属性：

```
<input type="checkbox" checked>
<dl compact>
```

checked和compact都是“最小化”属性。因为这些属性不要求指定值，所以声明然后使用这样的属性都非常简单。然而XHTML强制性地要求对使用的所有属性都必须指定值。对于“最小化”属性，可按下面的代码那样进行扩展：

```
<input type="checkbox" checked="checked">
<dl compact="compact">
<title>My sample page</title>
```

这是一个很小的差别，但这种统一性是保证代码有效所必需的。

1.2.2　从结构提取样式

许多标准把“样式从结构分离”鼓吹为在站点设计中采用CSS的主要好处——我同意这种观点，但有个小的限制。在后面将看到，无法真正分离标记和样式表。改变前者的结构，后者中的许多规则就变得没用了。

由于标记和CSS在很大程度上是相互关联的，因此可以认为样式是从结构中提取的。标记主要用于描述内容，但是，它常常包含一定程度的表现信息。然而，程度大小完全取决于设计人员。如果设计人员如此选择，就可以很容易地把表现方面的工作放到XHTML中——XHTML中充满了font、table和透明的GIF图形。

另一方面，样式表中可以包含规则，规则决定页面设计的许多方面：颜色、版式、图像、乃至布局等。如果把这些规则放在外部的样式表文件中并在站点页面中引用，就可以对站点中成千上万HTML文档的可视化显示进行控制。这不仅在日常的站点更新中有非常吸引人的前景，而且也利于今后对整个网站进行重新设计。在集中的样式表中简单修改很少几行就能对标记的表现进行很好的控制。

您和您的营销部门对此应该感到非常满意。

由于CSS可以保存在不同的文件中，因此在访问站点的第一个页面后，至少在表面上可以把整个站点的用户界面缓存起来。而在站点设计的标签汤时代，用户不得不为站点的每一页面重复下载大量的标记：嵌套的<table>元素、透明的GIF图形、<font>元素、bgcolor声明，以及对每个Web页面都要保持相同设计样式的标记。现在，一旦用户完全下载了一个站点的样式表，就可以很快地浏览该站点的其他页面，因为此时需要下载的标记已经很少了。

您的用户对此也应该感到非常满意。

最后，一个最明显的好处是：标记变得非常简单。这可进一步减少对用户的带宽要求，而下载页面的速度更快。当然，在您承担的任何一个搜索引擎优化项目中，网站都可从中受益。搜索引擎对现在已经变得很小的标记中的内容进行检索将非常便捷。

1. 避免过度使用div和class

为了使标记更少而抛弃表格，可用大量的class属性来取代常用的font元素，这种做法对初级开发人员来说并不少见。可能要处理像下列这样的导航条表格：

```
<!-- outer table -->
<table bgcolor="#000000" border="0" cellspacing="0" cellpadding="0">
<tbody>
<tr>
<td>
<!-- inner table -->
<table border="0" cellspacing="1" cellpadding="3">
<tbody>
<tr>
<td bgcolor="#DDDDDD"><font face="Verdana, Geneva, Helvetica, sans-
serif" size="2"><a href="home.html">Home</a></font></td>
<td bgcolor="#DDDDDD"><font face="Verdana, Geneva, Helvetica, sans-
serif" size="2"><a href="about.html">About Us</a></font></td>
<td bgcolor="#DDDDDD"><font face="Verdana, Geneva, Helvetica, sans-
serif"size="2"><a href="products.html">Our Products</a></font></td>
</tr>
</tbody>
</table>
<!-- END inner table -->
</td>
</tr>
</tbody>
</table>
<!-- END outer table -->
```

下列版本并不是很好：

```
<!-- outer table -->
<table class="bg-black" border="0" cellspacing="0" cellpadding="0">
<tbody>
<tr>
<td>
<!-- inner table -->
<table border="0" cellspacing="1" cellpadding="3">
<tbody>
<tr>
<td class="bg-gray"><a href="home.html" class="innerlink">Home</a></td>
<td class="bg-gray"><a href="about.html" class="innerlink">About Us
        </a></td>
<td class="bg-gray"><a href="products.html" class="innerlink">Our
        Products</a></td>
</tr>
</tbody>
```

```
</table>
<!-- END inner table -->
</td>
</tr>
</tbody>
</table>
<!-- END outer table -->
```

这就是所谓的classitis，是由设计师designer Jeffrey Zeldman(http://www.zeldman.com)创造的术语，用于描述在标记中充斥着过度使用class属性的情形。这会带来另外的问题。该例是用class属性，而不是在标记中清楚明白地表现最终目标来达到同样目的。所要做的改变是把所有bgcolor属性和font元素的值放在一个外部样式表中——这是一个良好开端，但标记仍然繁多。

更坏的是人们很容易屈从于divitis，即标记中充斥着过多的div元素。

```
<div id="outer-table">
 <div id="inner-table">
  <div class="innerlink"><span class="link"><a href="home.html"
class="innerlink">Home</a></span></div>
  <div class="innerlink"><span class="link"><a href="about.html"
class="innerlink">About Us</a></span></div>
  <div class="innerlink"><span class="link"><a href="products.html"
class="innerlink">Our Products</a></span></div>
</div>
</div>
```

您应该有足够的耐心召集大家并给大家解释。这里没有要写出良好架构标记的明显尝试。div和span是优秀的标记工具，但过分依赖它们则会导致代码臃肿、难于阅读。这不仅对您来讲是晦涩难懂的，对您的读者更是如此。还记得先前遇到的纯文本屏幕的例子吗(如图1-3所示)？如果隐藏了样式表并使用通用的标记，对非图形环境的用户来讲，要了解内容的上下文是非常困难的。

2. 转向良好定义的标记

另一种选择是使用意有所指的标记——只在不能用其他元素的地方使用div和span。本节将介绍一些策略，运用这些策略可把页面的标记修改为良好架构、良好定义的最小集，为介绍以后各章中的CSS技巧和策略做准备。

下面我们回顾一下在哈佛大学主页(如图1-1所示)中所用的一些HTML元素。我们将把一些良好架构的思想应用在老式的标记中，看看能否产生更好的标记。

3. 熟悉其他标记元素

在这里我们将进行内容描述而不是设计。因此对XHTML规范越精通，就越能用它准确地描述站点的结构。

更好地了解标题

在右边一栏中，我们来看看用于标记其中一个新闻项的标记：

```
<b>' Illuminating the beauty of life' </b>
<br>Yannatos starts 41st year conducting Harvard-Radcliffe Orchestra
<a href="http://www.news.harvard.edu/gazette/"><font size="-
1"><b>More</b></font></a><font size="-1"><br></font>
</td>
<td width="120">
<a href="http://www.news.harvard.edu/gazetta/"><img src="images/041029a.
jpg:alt="James Yannatos" border="0" height="120" width="120"></a><br><br>
```

在这个新闻项中，其内容是头条新闻的标题。然而从标记并不知道这就是标题：

```
<b>' Illuminating the beauty of life' </b>
```

对于使用桌面型浏览器的拥有正常视力的用户而言，除了用标记描述元素的外观外，为什么不用标题元素呢？

```
<h4>' Illuminating the beauty of life' </h4>
```

此代码使用了一个h4元素。如果在本文档层次结构中，在该标题之上还有三级标题，则使用h4是合适的。现在任何一个“用户代理”在浏览该页时，都能把此文本作为标题，并以自己最有效的方式显示它。

说明：

在构建良好定义的标记时，认为文档符合某种大纲是很有帮助的——将最重要的标题放在最上一级h1，在它之下的是一些h2级的标题，在每个h2级的标题之下是一个或两个h3级的标题，一直可到h6级标题。如何构想文档的大纲完全取决于您自己。可以建立一个对您来说有意义的模型，并在整个站点标记中保持一致。

更好地了解段落

在新闻故事的大字标题之后是一个长段落节选——但标记又一次掩盖了内容的意图：

```
<br>Yannatos starts 41st year conducting Harvard-Radcliffe Orchestra
<a href="http://www.news.harvard.edu/gazette/"><font size="-1"><b>More
</b></font></a><font size="-1"><br></font>
```

如果这是一个段落，应该使用下面的标记，不应仅依赖于换行元素(
)把它从周围的内容中分开：

```
<p>Yannatos starts 41st year conducting Harvard-Radcliffe Orchestra <a
title="Harvard Gazette: &quot;Illuminating the beauty of life&quot;"
href="http://www.news.harvard.edu/gazette/">More</a></p>
```

我们可以看到链接More在段落内——在本章后面部分将看到这一点，我们可以用CSS很容易地把这个锚移到下一行。因此，如果有意要把该锚保留在段落内，可以选择采用不同的方案。

更好地了解无序列表

站点的导航实际上就是一个链接列表。可以用无序列表替换表格形式的导航菜单，每个列表项包含一个到站点某个部分的链接，如下面的例子所示：

```
<ul>
<li><a href="home.html">Home</a></li>
<li><a href="about.html">About Us</a></li>
<li><a href="products.html">Our Products</a></li>
</ul>
```

说明：

由于JavaScript的小小魔力，内嵌的无序列表可以很快、很容易地转换为动态下拉菜单。最流行的一个例子是“Son of Suckerfish”(SoS)菜单脚本(http://www.htmldog.com/articles/suckerfish/dropdowns)，这是由设计人员Patrick Griffiths 和Dan Webb开发的。行为(JavaScript)和样式(CSS)可以分层形成良构标记的顶层基础，SoS菜单就是这样的完美例子，所有这些都非常好地降级为非CSS和非JS的Internet设备。

4. 注重内容而不是图形

首先，应该完成页面内容区域的清单。我们再考察一下哈佛大学主页(如图1-4所示)，

可以看到该页面布局可分解为以下几个部分：

- 标题栏
- 导航栏
  - 主导航栏
  - 次导航栏
- 内容
  - 主要内容
  - 次要内容
- 页脚导航栏
- 版权信息

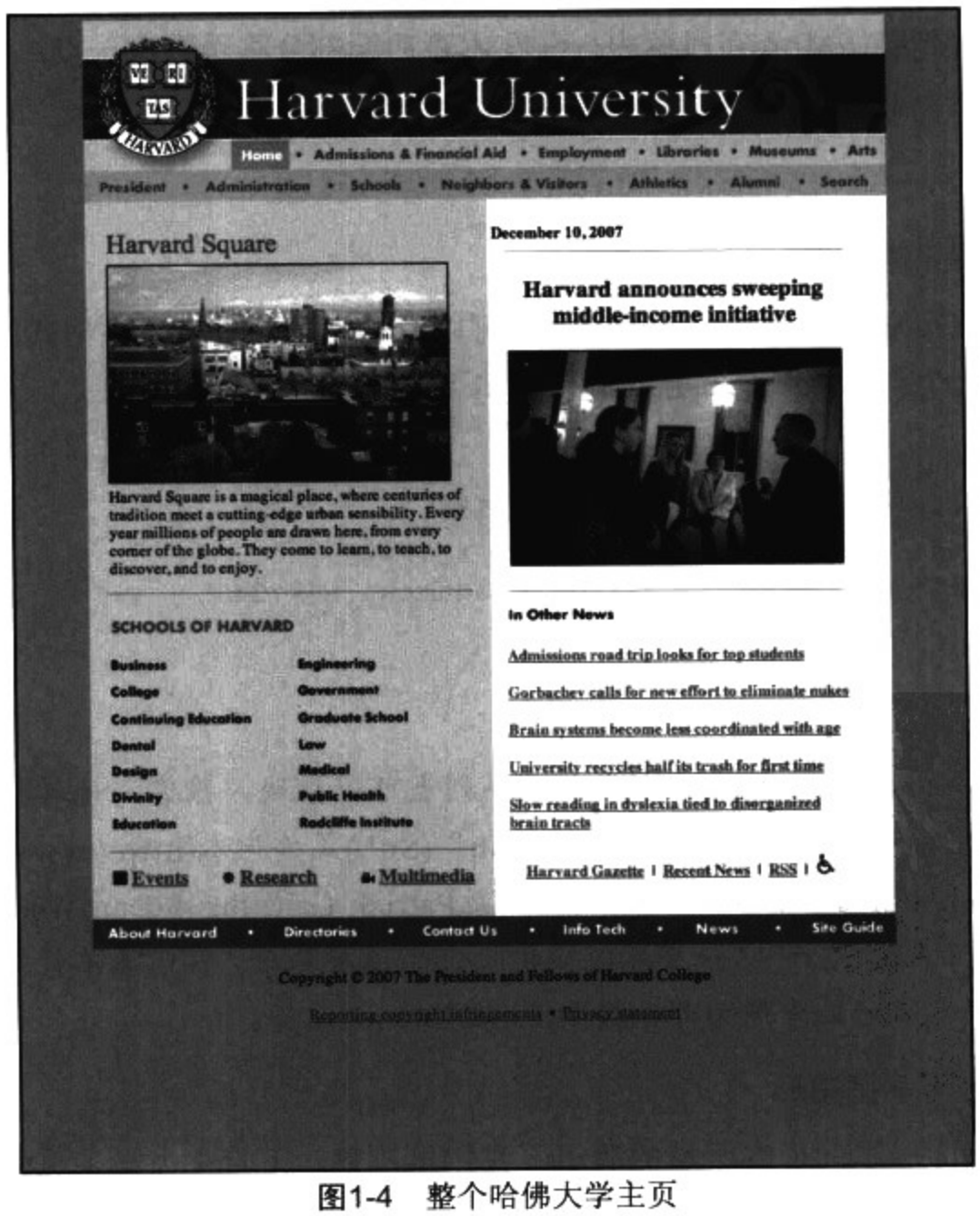


图1-4　整个哈佛大学主页

在两个栏内，进一步描述了不同的内容块，如图1-5所示。主内容栏为哈佛简史、社团链接列表和其他杂项链接列表。第二栏主要显示新闻，第一块是新闻特写，第二块是重要性稍低的新闻项(如图1-4所示)。

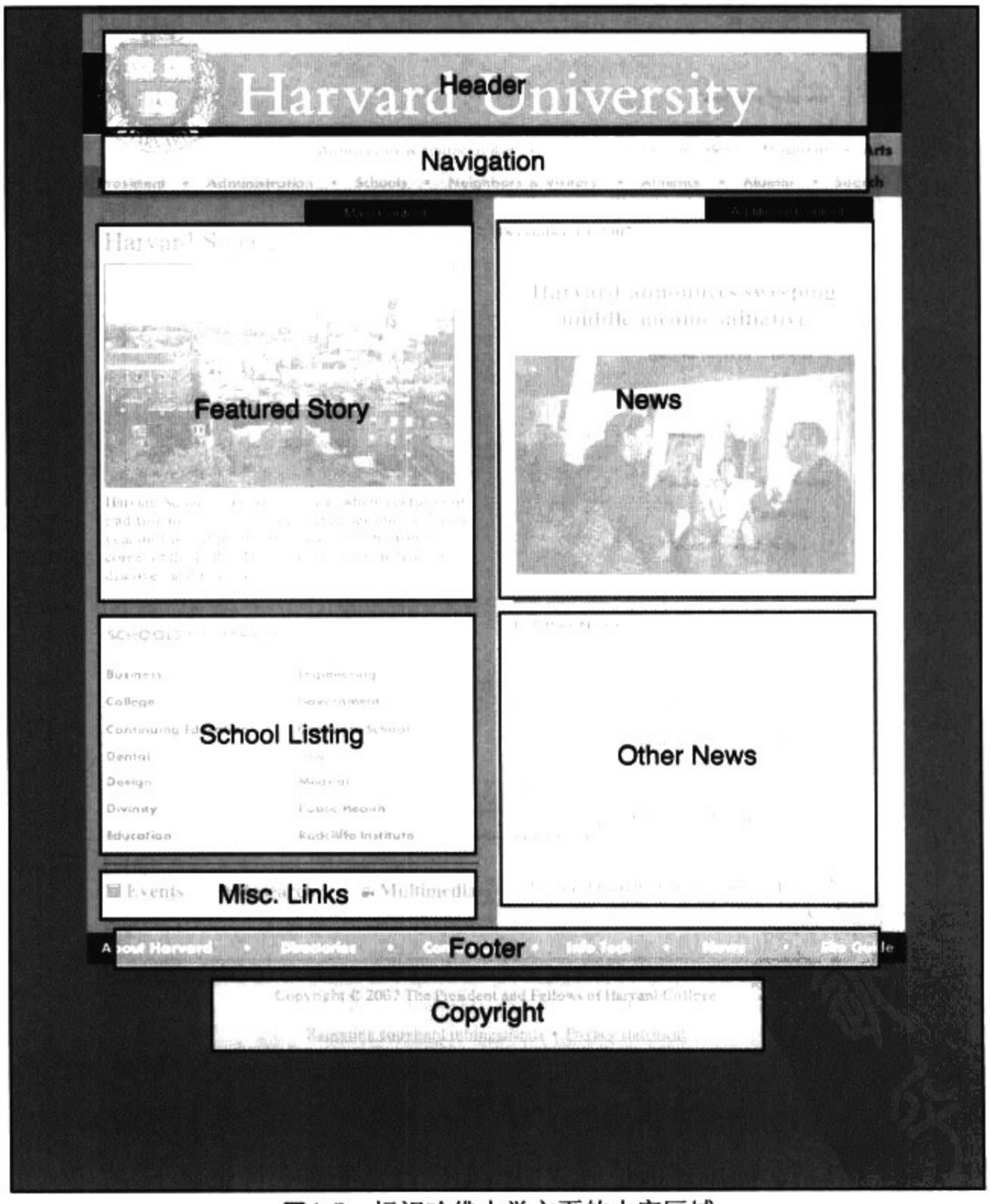


图1-5　标识哈佛大学主页的内容区域

从该页面内容的意境地图，就可通过标记对它进行描述。每一个内容区可用一个带唯一id的div描述，如有必要可在div中嵌套子内容块(在这里是两个内容栏的情形)。完成后可

得到如图1-6所示的XHTML文档。每个内容区都用一个div(带一个描述性的id)标记。嵌套顺序反映内容清单的大纲关系(如图1-5所示)。

图1-6　新XHTML模板大纲

在这里，标记基本上是一块白板，可在它上面对样式进行分层。用户并看不到样式，留下的是良构的、容易遵循的结构(如图1-7所示)。在本章其余部分以及整本书中都可对这些策略进行检验，看看如何迅速地把表现层添加到基本的有效标记之上。

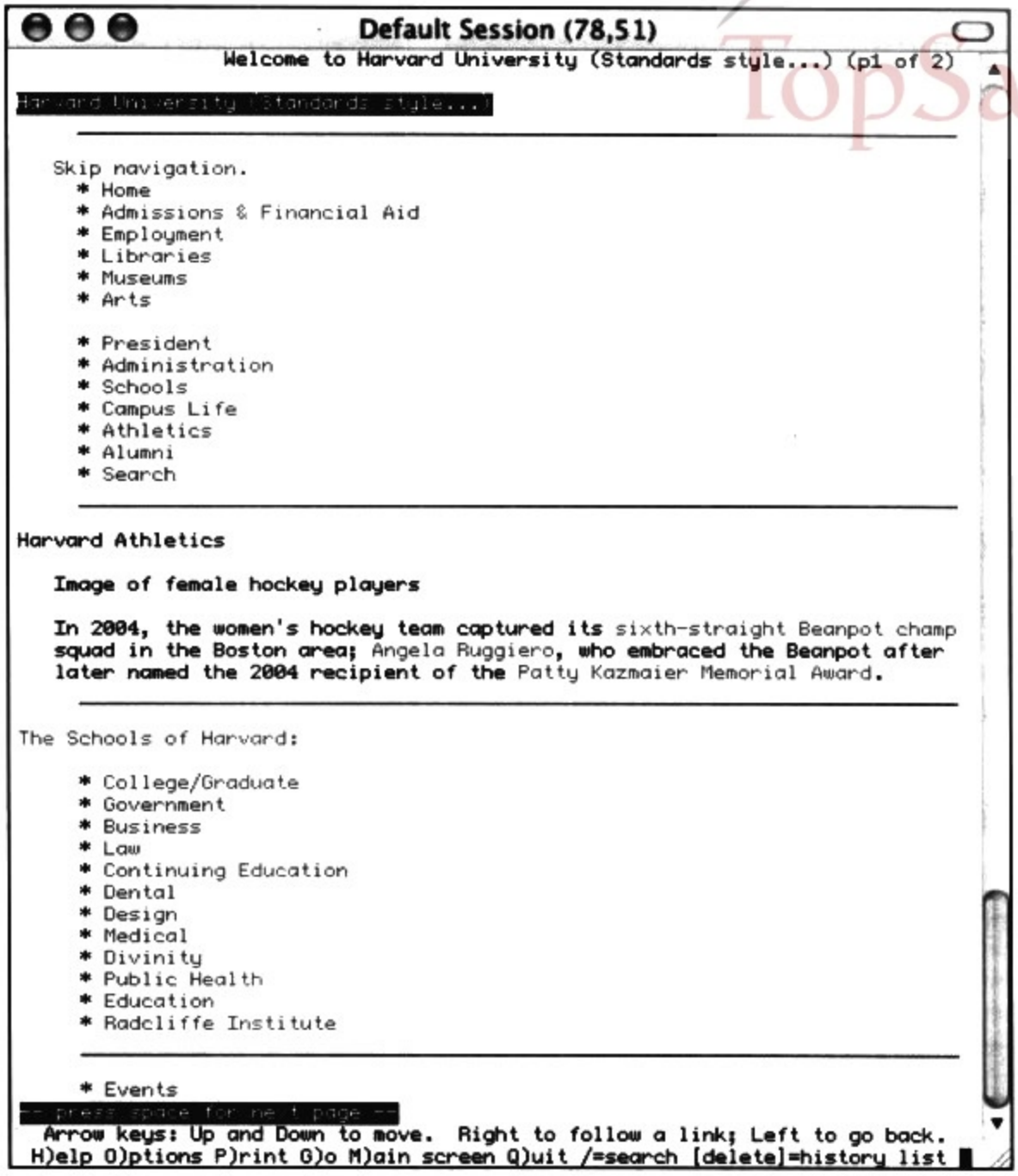


图1-7　修改后的哈佛XHTML页面在Lynx中的视图

1.3　CSS：添加样式层

和任何其他语言一样，要精通CSS就必须了解它的语法。这样不仅能熟练地编写样式表，而且还能充分了解浏览器是如何解释所编写的规则的。

1.3.1　更好地了解选择符

CSS由样式规则组成，样式规则由浏览器解释，然后应用到文档中的相关元素。请看

如图1-8所示的范例。

图1-8 一个简单的样式规则范例

每一个CSS样式规则都由两部分组成：一个选择符和一个声明块。选择符告诉浏览器该规则所影响的元素。声明块决定将要修改元素的哪些属性。从图1-8可以发现，选择符包括左花括号({)前的所有字符，但不包含左花括号。

声明块是规则的实体，由大括号括起来。声明块由一个或多个声明组成，声明的“属性/值”对决定了选择符中元素的样式。在图1-8中，color是属性，#36C是该属性的值。

看起来似乎很简单，事实也是如此。常识性逻辑遍布CSS语法。但这仅仅是开始。您可以用这些最简单的原则构建更加复杂的规则，从而对站点的表现进行很好的控制。

1. 类型选择符

我们再来回顾一下简单的h1规则：

```
h1 {
color: #36C;
)
```

这就是所谓的类型选择符，因为它指示浏览器选择指定类型的所有元素(在这里是标记中的所有h1)并以天蓝色显示。

您可能对这个规则感到奇怪，其实有很多font元素都采用这种形式。

说明：

是否对奇怪的color值#36C感到惊讶？这是十六进制颜色表示的一种缩写方式。如果RGB三色的每个十六进制对都相同的话就可这样缩写。所以把#3366CC写成#36C,，因为十六进制对红色(33)、绿(66)和兰(CC)都是相同的字符。同样可把#FFFF00缩写为#FF0，把#000000缩写为＃000等。

2. 通配选择符

另一种广泛使用的选择符称为通配选择符，它比类型选择符有更广的使用范围。除了选择指定类型的元素，通配选择符可以很简单地与任何类型的元素名匹配。其符号是星号或通配符，如下所示：

```
* {
color: #000;
)
```

该规则把文档的所有内容显示为黑色。相当简单，对吗？可能很少使用这样的规则，因为通配选择符的使用受一定范围的限制。更进一步讲，由于可测试方面的原因，存在是否支持通配选择符的问题。然而可以研究这种选择符的一些特定用法，有时仅留下这种记号而根本不显示。

3. 后代选择符

假设在某个时候遇到下列标记：

```
<p>I just <em>love</em> emphasis!</p>
<ul>
  <li>Don't <em>you</em>?!</li>
 <li>Oh, certainly.
  <ol>
    <li>I <em>still</em> love it!</li>
  </ol>
 </li>
</ul>
```

大多数浏览器都默认把em元素呈现为斜体。但您是否觉得有些武断？假设您希望所有在ul内的em元素都显示为大写的。利用目前所学知识，可以用类型选择符写一个规则并匹配所有em元素：

```
em {
  text-transform: uppercase;
}
```

但是我们只是希望匹配ul中的em元素——也就是说ul之外的em元素不受该规则影响。仅仅在选择符中用em匹配文档中的所有em，所以需要做小小的限定：

```
ul em {
  text-transform: uppercase;
}
```

该规则以一个后代选择符开始，告诉浏览器“选择ul元素中的所有em元素”。这正如父母的孩子是祖父母和曾祖父母的后代一样。按这种方式，该规则可被应用到该无序列表的每一层em元素——即使是包含在其有序列表内的em也是如此。最重要的是，这条规则不会被应用到起始标记p中的em元素，这正是我们所希望的。

我们承认，这种情况不会经常发生。但要记住，这种类型的选择符提供了对页面设计进行大粒度控制的手段。如果需要对内容中的特定部分免除全局样式规则的控制，现在我们有办法实现这一点。

4. 类选择符

还有更细粒度的控制吗？样式表可对标记中的各个方面进行控制。还记得在哈佛大学主页更新的时候用的class属性吗？它用来表示div的一个“项”分类。好了，现在该介绍类选择符了：

```
input.box {
 border: 1px solid #C00;
}
```

这个选择符允许CSS的作者选择元素，这些元素的class属性包含句点(.)后指定的值。在下面的示例中，制定的规则将选择class属性中包含字“text”的所有input元素，如下所示：

```
<form id="sample" action="blah.html" method="post">
  <fieldset>
 <p>
  <label for="box-one">Box #1:</label>
  <input type="text" id="box-one" size="15" class="text" />
 </p>
 <p>
  <label for="box-two">Box #2:</label>
  <input type="text" id="box-two" size="15" class="text" />
 </p>
 <input type="submit" id="submit" value="Submit!" />
 </fieldset>
</form>
```

该示例表单最后的submit按钮不受影响，两个文本字段(class属性值为“text”)将受规则影响。

如果希望class选择符更通用，可简单地省略“input”，如下所示：

```
.text {
 border: 1px solid #C00;
}
```

虽然您可能没注意到，但通配选择符在这里出现了。如果愿意，可把这条规则写成如下形式：

```
*.box {
 border: 1px solid #C00;
}
```

这两个选择符可达到同样的目的：都选择class属性中包含字“text”的所有元素。

5. id选择符

与类选择符类似，id选择符可以基于id属性选择一个元素：

```
h1#page-title {
 text-align: right;
}
```

但是class选择符用句点(.)，而id选择符用井号(#)。根据这条规则，将选择id属性值与井号后文本(名为“page-title”)匹配的h1元素。和class选择符一样，我们也可以使用隐含的通配选择符，即把井号前的h1去掉即可：

```
#page-title {
  text-align: right;
}
```

如何做到这一点？这是因为一个元素的id属性值在有效XHTML文档的上下文中是唯一的——在我们的标记中没有别的元素与h1共享id“page-title”。因此我们知道两个规则匹配的是标记中唯一的一个元素，因此这两个规则是等价的。

当id选择符作为后代选择符的基础(见下面的示例)时，它们的威力才显现出来：

```
#content h2 {
  text-transform: uppercase;

}
```

像这样的规则是复杂布局的基础。这就使CSS设计人员能构建完全与上下文相关的样式规则。例如，上面的规则将只选择id为“content”的所有h2后代元素。文档中所有其他二级标题元素则不受该规则的影响。

1.3.2　其他选择符

这一节我们将介绍规范中其他可用的选择符。

说明：

在本书写作时，Microsoft Internet Explorer 6 for Windows (MSIE 6/Win)还不支持这些选择符。其他流行的浏览器(包括Internet Explorer 7 (MSIE 7/Win)、Firefox、Opera和Safari)则完全支持这些选择符，其中许多选择符对CSS 2来讲是新的。MSIE 6/Win在实现这个规范时力求尽量简单。

MSIE 7/Win完全采用了这些选择符，彻底消除了支持前面版本的要求，最终消除了必须检查站点登录文件的要求。检查站点登录文件是为了看看在CSS驱动的设计中实现这些选择符之前是否要支持MSIE 6/Win和其他老式浏览器。

1. 子选择符

我们已经发现，我们所写的规则可以立即影响另一元素之下的所有元素。假定我们要对文档中body内的所有段落进行样式控制：

```
<body>
 <p>I am the very model...</p>
 <div class="news">
  <p>Of a modern markup general.</p>
 </div>
 <p>I use every attribute, be it vegetable or mineral.</p>
</body>
```

为了使所有段落以粗体显示，可以选择下列两个规则中的一个。

```
p {
  font-weight: bold;
}
```

或：

```
body p {
  font-weight: bold;
}
```

上述两个规则的效果相同。前面已经讲到，它们将选择文档中body元素内任一层的所有段落。

但是，如果希望对匹配范围进行一些限制，且希望只选择文档层中恰好是body元素下的段落，那该如何实现呢？

现在介绍子选择符：

```
body>p {
  font-weight: bold;
}
```

大于符号(>)指示用户代理选择子一级的所有p元素而不是所有后代。因此包含在div内的段落不受影响，因为它们不是body元素的子元素。当然div前后的段落元素马上就会受

到影响，因为它们都是body元素的子元素。

2. 属性选择符

为了在表格中不使用class属性，CSS规范定义了属性选择符。这些选择符不仅能匹配给定元素的class属性，也可以匹配它拥有的任何其他属性。属性选择符用一对方括号表示，可按表1-1中的四种方法的任意一种进行匹配。

表1-1　四种方法

属性选择符语法	结　果
[x]	当元素有指定的x属性时进行匹配，不管属性值
[x=y]	当元素的x属性值正好等于y时进行匹配
[x~=y]	当元素的x属性值是一个以空格为定界符的字列表，且其中之一正好是y时进行匹配
[x|=y]	当元素的x属性值是一个以y开头的、以连字符为分隔符的字列表时进行匹配

似乎还有点不明白。下面给出一些具体示例对属性选择符以及它所选择的元素进行了说明，如表1-2所示。

表1-2　进一步说明

选 择 符	意　义	选择的元素	不选择的元素
p[lang]	选择具有lang属性的所有段落元素	<p lang="eng"> <p lang="five">	<p class="lang"> <p>
p[lang="fr"]	选择lang属性值正好为“fr”的所有段落元素	<p lang="fr"> <p class="gazette" lang="fr">	<p lang="fr-Canada"> <p lang="french">
p[lang~="fr"]	选择lang属性值包含“fr”的所有段落元素	<p lang="fr"> <p lang="en fr"> <p lang="la sp fr">	<p lang="fr-Canada"> <p lang="french">
p[lang|="en"]	选择lang属性值正好包含“en”或以“en-”开头的所有段落元素	<p lang="en"> <p lang="en-US"> <p lang="en-cockney">	<p lang="US-en"> <p lang="eng">

至少可以说，这些选择符的潜在应用是非常令人激动的。再回顾一下前面介绍过的表格示例，现在就变得简单了：

```
<form id="sample" action="blah.html" method="post">
 <fieldset>
```

```
    <p>
     <label for="text-one">Box #1:</label>
     <input type="text" id="text-one" size="15" />
    </p>
    <p>
     <label for="text-two">Box #2:</label>
     <input type="text" id="text-two" size="15" />
    </p>
    <input type="submit" id="submit" value="Submit!" />
   </fieldset>
</form>
```

利用属性选择符，可以很容易锁定type属性值正好为“text”的所有input元素：

```
input[ type="text"] {
  border: 1px solid #C00;
}
```

这个方法的好处是<input type="submit" />元素不受影响，这个边框只应用于所期望的文本字段。不必再把标记与用于表现的class属性混在一起。取而代之的是，可把XHTML本身作为样式表的一类API，编写一些选择符使之与文档的结构相对应。

说明：

为了消除对类选择符的偏见，我再次向大家保证我并不否认语义的重要地位。目前所能使用的选择符 (也就是与MSIE /Win兼容的选择符)虽然可能不是理想的工具，但都是优秀的工具。即使在表格中使用类不是最完美的语义解决方案，但类可提供很大的灵活性，并在目前的设计中可对结构进行控制。

1.3.3 多重声明组合

迄今为止我们看到的所有样式规则都只有一个声明——令人欣慰的是并不是必须这样。如果语法必须是下面这样的，想像一下您的CSS该是多么冗长。

```
h1 { color: #36C; }
h1 { font-weight: normal; }
h1 { letter-spacing: .4em; }
h1 { margin-bottom: 1em; }
h1 ( text-transform: lowercase; )
```

如果浏览器大声朗读您的样式表，这一小段或许就是治疗失眠症的最好良方。听起来应该是这样的：“选择所有的h1元素并把它们的颜色设置为#36C。选择所有的h1元素

并把它们的字重设置为正常类型。选择所有的h1元素并把字符间距设置为0.4个字符宽度(em)。选择所有的h1元素并把下边界设置为1个字符宽度。选择所有的h1元素并把文本都转换为小写。”

值得庆幸的是，有方法可以解决这种过分的冗长和混乱的问题。同一个选择符的多重声明可压缩为一个以分号隔开的、易于实现的规则。下面我们再来考察一下据此修改后的h1规则：

```
h1 {
 color: #36C;
 font-weight: normal;
 letter-spacing: .4em;
 margin-bottom: lem;
 text-transform: lowercase;
}
```

这个示例表明，可以把h1元素的多个属性放在一个规则中。这就可以保持样式表的简洁，易于维护。总之，简洁是样式之魂。

说明：

当用多重声明编写规则时，在技术上分号被认为是一个定界符——用于把前一声明的结尾和另一声明的开始分开。因此在样式块中省略最后一个规则的分号也是完全有效的，因为没有后续的声明。为了一致起见，建议在每个声明后面都加上分号。这样，当需要改变规则中某个声明的位置或全面编辑一个属性值时，就不用操心分号的问题。

1.3.4　对选择符进行分组

如果要把h1规则的样式应用到文档中的其他元素又该如何实现呢？如果希望所有h2和h3元素都共享h1的样式，当然可以显式地进行声明：

```
h1 {
  color: #36C;
  font-weight: normal;
  letter-spacing: .4em;
  margin-bottom: lem;
  text-transform: lowercase;
}
h2 {
  color: #36C;
```

```
    font-weight: normal;
    letter-spacing: .4em;
    margin-bottom: 1em;
    text-transform: lowercase;
}
h3 {
    color: #36C;
    font-weight: normal;
    letter-spacing: .4em;
    margin-bottom: 1em;
    text-transform: lowercase;
}
```

您或许会说，代码太长了。规范再次提供了另外一种方法使CSS变短，以降低对带宽的要求。即，当几个规则共享相同的声明时，可把这些选择符聚合到一个以逗号分隔的列表中。例如，可把上述三个标题规则写到一个规则中：

```
h1, h2, h3 {
    color: #36C;
    font-weight: normal;
    letter-spacing: .4em;
    margin-bottom: 1em;
    text-transform: lowercase;
}
```

列表顺序是不相关的。这些声明将应用于选择符中的所有元素。

当然，还可以对共享属性进一步压缩。如果规则只是共享某些属性，只需为共享值创建一个组规则，而为不共享的属性创建独立的规则。如：

```
#content {
 border: 1px solid #C00;
 padding: 10px;
 width: 500px;
}
#footer {
 padding: 10px;
 width: 500px;
}
#supplement {
 border: 1px solid #C00;
 padding: 10px;
 position: absolute;
```

```
 left: 510px;
 width: 200px;
}
```

可以看到，上述规则都共享一个值为10px的padding属性。规则#content 和#footer共享相同的width属性。#content 和 #supplement共享一个1px的红色边框。根据同样的思想，可以把值相同的属性放在组选择符中，如下所示：

```
#content, #footer, #supplement {
 padding: 10px;
}
#content, #footer {
 width: 500px;
}
#content, #supplement {
 border: 1px solid #C00;
}
#supplement {
 position: absolute;
 left: 510px;
 width: 200px;
}
```

看起来效果不是很明显——修改之前是16行，现在是14行，还引入了额外的规则。但是这种智能分组的优势是累积的，一旦要编写更复杂的样式规则，这种优势就非常明显。当把共享样式声明压缩后，只需修改一个样式规则就可改变#content和#supplement元素的边框颜色。一旦样式表不再是几十行的长度而是几百行的长度，在需要花时间修改共享值时，分组方法就会非常省时间。

1.3.5　继承

在编写CSS规则时，我们应该还记得一些属性(和给它们赋的值)是可由后代元素继承的。事实上，这与我们从家庭继承遗产是很类似的。事实上，一个Web页面并不是与其他页面完全不同。在父-子关系中，元素从包含它们的元素中继承样式属性的元素。事实上，完全可以画出一棵页面元素的家族树，如图1-9所示。

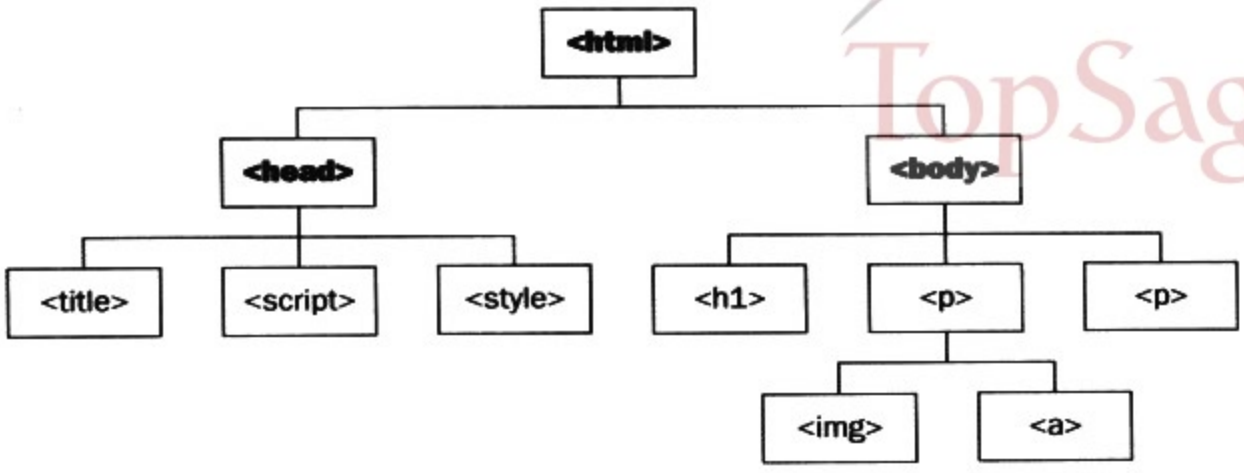


图1-9　标记文档树

1. 检查元素的层次

html元素是页面的根元素，这是“文档树”的基础。它是所有子元素(即head和body元素)的父元素，head和body元素有自己的子元素，它们的子元素又有自己的子元素，等等。有相同父元素的元素被称为兄弟元素。从树上某个点，向下超过一层的标签称为后代元素。反过来可以认为，接近顶层的元素是更接近底层的元素的祖先。这里我们得到一棵层次分明的家族树，规律性稍差些。

考虑这种层次结构中的标记，就很容易想像到样式是如何向下传播到不同分枝的。考察下面的示例：

```
body {
 color: #000;
 font-family: Georgia, "Times New Roman", serif;
 font-size: 12px;
}
```

根据前面讲到的语法规则，这条规则告诉用户代理选择所有的body元素并把body内的字体设置为serif(可在用户机器上按顺序找找字体Georgia、Times New Roman或通用的sans-serif font)、字号设置为12px、黑色(#000)。

现在把继承规则应用于文档，这三个属性将传到body内的所有元素——或用文档家族树的语言，传到body的所有后代元素。我们已经看到了为什么CSS具有这样强大的表现机制。一个单一的四行样式规则对所有font元素发生了作用。

继承或许是CSS最强大的地方，但有时也更容易出现混乱。值得注意的是，不是所有属性都可以继承。外边距和内边距就是两个例外。这些属性只能单独应用于某个元素，而不能被其后代继承。

关于什么属性可被元素的后代继承，最好参考规范本身(http://www.w3.org/TR/CSS21/about.html#property-defs)。

2. 重写继承

如果不希望文档的特定部分从祖先继承属性，又该如何呢？考察下列示例：

```
<body>
<p>I still like big blocks.</p>
<ul>
<li>...but lists are even cooler.</li>
</ul>
</body>
```

如果把前面针对body元素的CSS规则应用在这里，则p和li内的所有文本都将继承在那里声明的属性值。但是用户要求所有列表项都要为红色(他妻子偏爱的颜色)的sans-serif字体，您该怎么办呢？只需写一条规则就可选择您所关心的后代元素，如下所示：

```
body {
 color: #000;
 font-family: Georgia, "Times New Roman", serif;
 font-size: 10px;
}
li {
 color: #C00;
 font-family: Verdana, sans-serif;
}
```

在这里我们初次看到了“层叠样式表”的层叠部分是如何工作的。因为列表项是body元素的后代，第二条规则有效地打破了继承链并把声明的样式应用于所选的元素——在这里是li。又由于没有为列表项声明新的字体大小，因此它们仍然从祖先body继承该属性值(10px)。最终结果是，用户看到的列表项是按所要求的红色、sans-serif字体显示，而该页的其他元素则继承body的规则。希望您、您的用户、您用户的妻子都能对这个结果感到满意。

1.3.6　综合应用

我们再来研究一个过度使用标记的示例。考虑一个相当简单的导航条，如图1-10所示。

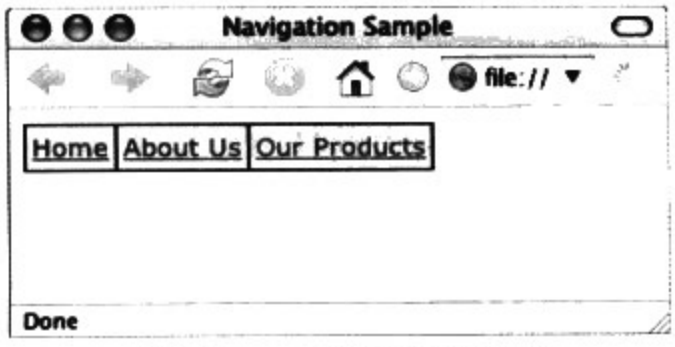


图1-10　一个简单的导航条

我们注意到这三个链接都是浅灰色的、水平排

列的、每个链接都有一个1px的黑色边框。一开始您可能觉得很简单，但在考虑如何用传统标记方法创建时才知道不是这么回事。

```
<!-- outer table -->
<table bgcolor="#000000" border="0" cellspacing="0" cellpadding="0">
<tbody>
<tr>
<td>
<!-- inner table -->
<table border="0" cellspacing="1" cellpadding="3">
<tbody>
<tr>
<td bgcolor="#DDDDDD"><font face="Verdana, Geneva, Helvetica, sans-
serif" size="2"><a href="home.html">Home</a></font></td>
<td bgcolor="#DDDDDD"><font face="Verdana, Geneva, Helvetica, sans-
serif" size="2"><a href="about.html">About Us</a></font></td>
<td bgcolor="#DDDDDD"><font face="Verdana, Geneva, Helvetica, sans-
serif" size="2"><a href="products.html">Our Products</a></font></td>
</tr>
</tbody>
</table>
<!-- END inner table -->
</td>
</tr>
</tbody>
</table>
<!-- END outer table -->
```

创建一个这样简单的导航条涉及两个表。第一个被称为外层表，其背景色为黑色(bgcolor="#000000")。内表没有自己的背景色，但是单元格之间的间距为1px。这就使得父表的黑色背景在创建每个导航项的“边框”时显现出来。内表还包括一个3px的单元格内边距，这样每个链接的文本和表格单元(<td>)之间有一定的空间。最后，内表每个单元格的背景色被设置为灰色(bgcolor="#DDDDDD")，还用一个<font>元素来指定合适的字体和大小。

24行代码(大约1KB数据)！对于一本讲Web设计的书籍，包含这样的代码片断似乎无伤大雅。但想像一下将这段代码传播到站点的二十多个(可能上百个，甚至上千个)页面将会怎样。当要求修改这段代码时又将会怎样？或许市场营销部门需要把灰色背景改为统一的浅绿色背景，或者把Arial字体设为标准化的字体。每一个要求都需要对用到这段标记的

若干个页面进行编辑、测试和重新配置，以满足需要。

这足以说明这完全不是令人满意的方案。令人欣慰的是，我们可以用所学的XHTML和CSS来改进这个导航条。首先引入一些新标记。我们从前面臃肿的、充斥着font的表中摆脱出来，来看看另一种不同的解决方案：

```
<ul id="nav">
<lia href="home.html">Home</a></li>
<li><a href="about.html">About Us</a></li>
<li><a href="products.html">Our Products</a></li>
</ul>
```

对了，这就是一个简单的无序列表。正如本章前面所讲的那样，我们不必关心这一层的表现形式。只需确保按正确的方式标记内容，与前面的标记一致。标记结果如图1-11所示。

首先，去掉不合时宜的圆点项目符号(bullet)。

```
ul#nav, ul#nav li {
 list-style: none;
 margin: 0;
 padding: 0;
}
```

结果如图1-12所示。

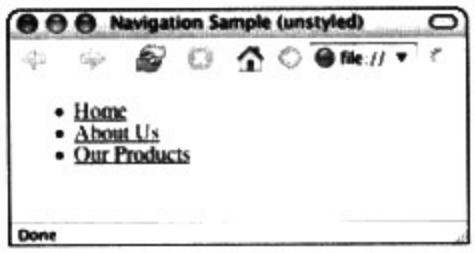


图1-11　一个简单的无序列表

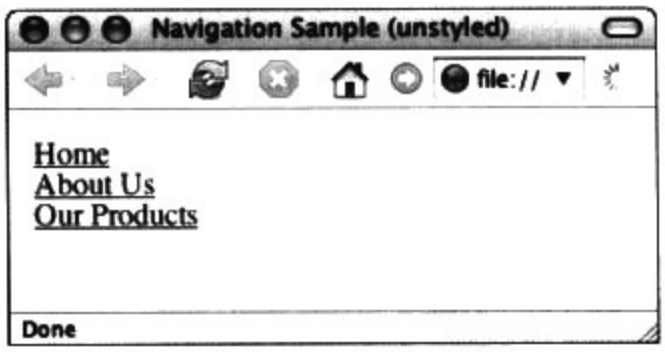


图1-12　没有圆点项目符号的无序列表

可以看到，在这里可以非常自由地使用ID选择符。通过在规则中指定ul#nav，就可以只把样式应用于导航列表(和它包含的元素)，而页面内的其他标记不受影响。通过把规则ul#nav和ul#nav li组合到一个选择符(以逗号为分隔符)，就可同时把列表项前的圆点去掉(list-style: none)，同时还把多余的外边距和内边距去掉了(margin: 0; padding: 0)。

当然最初的导航表是水平排列的，现在的列表却不是这样。然而这很容易修改。

```
ul#nav, ul#nav li {
 float: left;
 list-style: none;
 margin: 0;
 padding: 0;
}
```

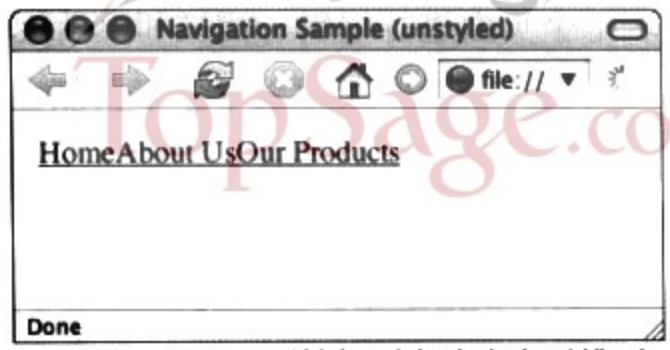


图1-13　用float属性把列表改为水平排列

结果如图1-13所示。

把float属性加到规则中，列表就可以按行排列。每个列表项漂浮在前一列表项的右边，把它们从原来的竖直排列改变为水平排列，如图1-13所示。

说明：

float模型是一个强大的CSS构件，事实上它是许多基于CSS布局的基础。Eric Meyer的文章*Containing Floats* (http://www.complexspiral.com/publications/containingfloats)更详细地介绍了float并纠正了关于这个工具的许多错误概念，使用这个工具是非常方便的。

如果对如何用float使列表居中排列感兴趣，可参考David Hopkins的文章*When Is a Float Not a Float?*(http://www.search-this.com/2007/09/19/when-is-a-float-not-a-float)。

在这个基本布局之上，可以添加更多的可视化组件。首先添加黑色边框和灰色背景：

```
ul#nav {
 font-family: Verdana, Geneva, Helvetica, sans-serif;
 font-size: .82em;
 background-color: #DDD;
}

ul#nav li a {
 border: 1px solid #000;
 display: block;
 float: left;
 padding: 3px;
}
```

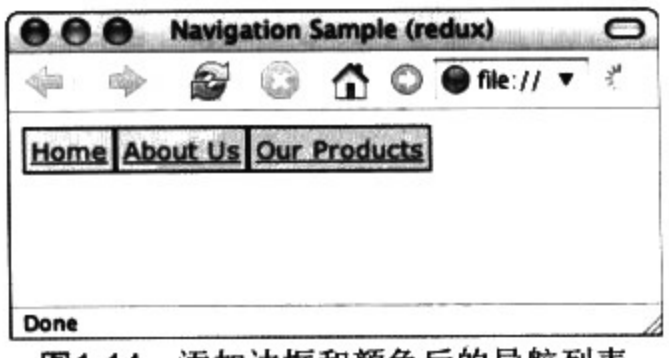


图1-14　添加边框和颜色后的导航列表

结果如图1-14所示。

可以看到，灰色背景色被直接应用到这个无序列表。利用继承性可以把sans-serif字体应用到其中的所有元素，也可以设置列表的font属性。锚被默认呈现为内联元素。这将引起一个小问题，因为任何内边距只影响它们的水平边。由于希望导航链接每边的内边距都为3px，这就必须把链接转换为块级项(display: block)。

然而事情还没结束。我们注意到Home和About Us 链接之间的边框以及About Us 和Our Products链接之间的边框太粗。这是因为每一项的边框与它的兄弟毗邻，当两个链接接触时造成双层边框。所以需要对样式做小小的改变：

```
ul#nav li a {
 border-color: #000;
 border-width: 1px 1px 1px 0;
 border-style: solid;
 display: block;
 float: left;
 padding: 3px;
}
```

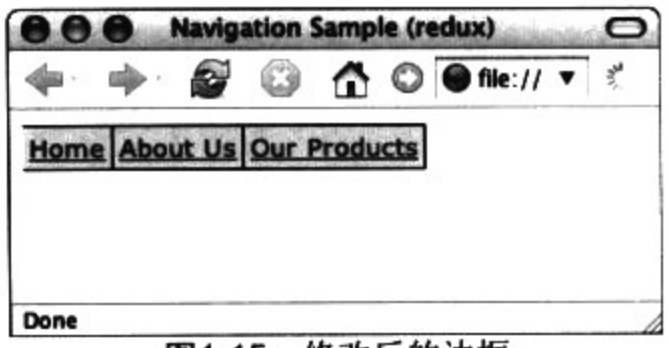


图1-15　修改后的边框

结果如图1-15所示。

边框声明越详细(border-color、border-width和border-style)，就越与以前的边框相同：1px solid #000，但是border-width声明中的最后一个0指示浏览器去掉每个列表项的左边框，而上、右和下边框宽度为1px。现在需要且只需要恢复第一个列表项的左边框——不要再造成双层边框。要实现这一点，就需要对标记中的第一个列表项应用class属性：

```
<ul id="nav">
 <li class="first"><a href="home.html">Home</a></li>
 <li><a href="about.html">About Us</a></li>
 <li><a href="products.html">Our Products</a></li>
</ul>
```

既然在文档结构中添加了一个“钩(hook)”，就可直接编写一个应用于且只应用于那个元素边框的样式规则：

```
ul#nav li.first a {
 border-width: 1px;
}
```

图1-16　最后的列表

结果如图1-16所示。

终于完成了！下面就是整个规则集：

```
ul#nav, ul#nav li {
 float: left;
 list-style: none;
 margin: 0;
 padding: 0;
}
```

```
ul#nav {
 font-family: Verdana, Geneva, Helvetica, sans-serif;
 font-size: .82em;
 background-color: #DDD;
}

ul#nav li a {
 border-color: #000;
 border-width: 1px 1px 1px 0;
 border-style: solid;
 display: block;
 float: left;
 padding: 3px;
}

ul#nav li.first a {
 border-width: 1px;
}
```

我们已成功地把分组、ID、类选择符和一些继承结合起来，把一个简陋的导航列表变成一个水平排列的导航菜单。我们不能满足于荣誉(虽然令人愉快)，而应该分析这种方案的优点。这种方法真的比表格好用吗？

绝对是这样！不得不承认，当把XHTML和CSS加在一起时，代码行数并没有明显减少。但是获得了从结构提取内容样式的方法。和表现代码与标记混合的方式相比，层叠样式表在用户界面方面做得更好。如果现在需要在菜单中添加另一个链接，只需在导航列表末尾简单添加另一个li，CSS就会处理其表现形式。

1.4 了解层叠

既然了解了CSS语法基础，我们将进一步研究CSS语法的机制，了解用户代理是如何确定应该给用户传递什么样的样式。

1.4.1 探寻样式来源

要探寻样式来源，必须首先找到合适的规则，因为样式规则的来源决定了它在层叠中的“影响力”。下面是样式表的三个方面来源：

- 用户代理——为了完全遵循CSS规范(http://www.w3.org/TR/CSS21/conform.

html#conformance)，在应用任何其他样式规则之前必须把一个默认样式表应用于文档。这个内部的CSS规则集为每个HTML元素创建了默认的显示规则。这样浏览器就知道如何用华丽的serif大字体显示h1或在每个无序列表项前添加一个圆点项目符号。

- 用户——用户也可以编写CSS。用户样式表被引进了CSS2规范，为用户重写由页面作者选择的字体和颜色提供了手段。一些设计人员或许对此感到茫然，但这确实是非常重要的动机。在一定的设计条件下，一些用户或许不能发现您的站点。编写定制的CSS可以使用户增加字体大小，这些字体可能是不合法的，或者是为了使用户避免一些特定的颜色/对比组合，这些颜色/对比组合对他们是不可见的。
- 作者——也就是您。这些是您的标记中包含的样式表，也是本书要介绍的主要内容。

当用户代理要从这三个方面的样式表中选择时，必须弄清楚把什么样的样式规则呈现给最终用户。为此必须为每个源分配一个确定的重要程度(或“权重”)。上述列表是按重要程度从低到高排列的(也就是说，浏览器默认样式规则的重要程度比用户的低一些，用户规则的重要程度又比作者在CSS文件中指定的规则低一些，我们知道这些CSS文件是驻留在站点服务器上的)。

然而作者和用户都可定义“!important”规则：

```
h2 {
 font-size: 2em !important;
 color: #C00 !important;
}
```

根据CSS规范(http://www.w3.org/TR/CSS21/cascade.html#important-rules)，“!important”规则使“作者样式表和用户样式表的优先级达成一个平衡”。前面已经提到，用户样式表所包含规则的权重远低于作者CSS所包含规则的权重。但是“!important”规则把这个关系颠倒过来了，即用户进行“!important”声明后，它的权重总比作者的规则高，如图1-17所示。

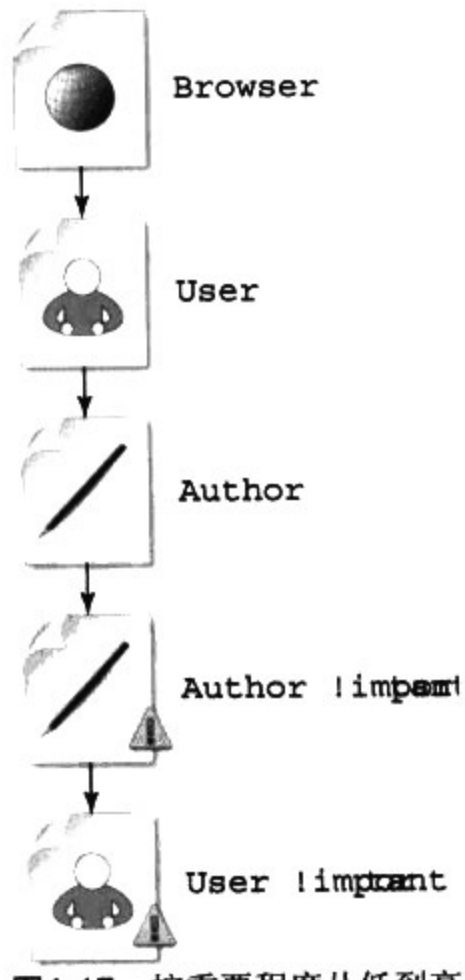


图1-17　按重要程度从低到高排列的各种样式

这是如何影响CSS的呢？我们来看看浏览器如何确定基本

段落元素(p)的样式。对所有可用的样式(浏览器的、用户的和作者的)进行解析后，对程序清单1-1、程序清单1-2和程序清单1-3的所有相关样式进行了评估。

程序清单1-1：浏览器的样式表

```
p {
 color: #000;
 font-size: 1em;
 margin: .9em;
}
```

程序清单1-2：用户的样式表

```
p {
 color: #060 !important;
}
```

程序清单1-3：作者的样式表

```
p {
 color: #300;
 font-size: 1.2em;
 line-height: 1.6em;
 padding: 10px;
}
```

暂时把用户的样式表放在一边。因此对一般用户，段落样式根据最后计算的规则确定，如下所示：

```
p {
 color: #300;           /* author overwrites browser rule */
 font-size: 1.2em;      /* author overwrites browser rule */
 line-height: 1.6em;    /* specified only by author */
 margin: .9em;          /* specified only by browser */
 padding: 10px;         /* specified only by author */
}
```

说明：

注意在上述代码中CSS注释的使用。在CSS中一对斜杠和星号内的内容就是注释。

现在，如果有用户用如程序清单1-2所示的用户样式表浏览页面，最后结果稍微有些变化，如下所示：

```
p {
color: #060;/* user !important rule overwrites author rule */
 font-size: 1.2em; /* author overwrites browser rule */
 line-height: 1.6em; /* specified only by author */
 margin: .9em; /* specified only by browser */
 padding: 10px; /* specified only by author */
}
```

1.4.2　根据优先级排序

我们对每个选择符的优先级评定了一个等级，这也是选择符重要性的另一个定性评估指标(http://www.w3.org/TR/CSS21/cascade.html#specificity)。一个规则的优先级越高，在浏览器对所有规则进行筛选时产生的影响也越大。例如，基于id的选择符，其优先级天生就比类驱动的选择符高，因为根据设计，id在每个文档只出现一次。

A: 优先级由选择符语法本身确定，并根据下列四个独立因子进行加权。

B: 该选择符是某元素的HTML样式属性还是一个真正选择符。

C: 选择符中id属性的数量。

选择符中其他属性(例如，[lang]、[rel]、[href])和伪类名(例如，:hover、:visited、:first-child)名数量。注意，类选择符(如li.active)是一种属性选择符，并被归为一类。

D: 选择符中其他元素(如a、li、p等)数量和伪元素(如 :before、:after等)名数量。

有了这四种因子，要计算某个选择符的重要程度就相对容易。表1-3是按优先级从低到高排列的选择符列表(列A~D)。

表1-3　选择符列表

选择符	A	B	C	D	优先级特性
a	0	0	0	1	0,0,0,1
h3 a	0	0	0	2	0,0,0,2
ul ol+li	0	0	0	3	0,0,0,3
ul ol li.red	0	0	1	3	0,0,1,3
li.red.level	0	0	2	1	0,0,2,1
#other-news	0	1	0	0	0,1,0,0
style=" ..."	1	0	0	0	1,0,0,0

有了这些信息，就可以计算前面的简单段落样式的优先级，如程序清单1-4、程序清

单1-5和程序清单1-6所示。

程序清单1-4：浏览器的样式表

```
p { color: #000; font-size: 1em; margin: .9em; }
/* A:0, B:0, C:1, D:1 = specificity of 0,0,0,1 */
```

程序清单1-5：用户的样式表

```
p { color: #060 !important; }
/* A:0, B:0, C:1, D:1 = specificity of 0,0,0,1 */
```

程序清单1-6：作者的样式表

```
p { color: #300; font-size: 1.2em; line-height: 1.6em; padding: 10px; }
/* A:0, B:0, C:1, D:1 = specificity of 0,0,0,1 */
p.gazette { color: #0C0; }
/* A:0, B:0, C:1, D:1 = specificity of 0,0,1,1 */
p#footer { color: #FFF; }
/* A:0, B:1, C:0, D:1 = specificity of 0,1,0,1 */
```

因为给颜色属性分配了多重规则，浏览器需要用优先级规则确定ID为p#footer的段落和含有类p.gazette的段落用什么颜色。

可以看出p#footer的优先级最高，其次是p.gazette。

假定站点访问者没有用户样式表(因此不受“!important”规则的影响)：

(1) id为footer的段落元素显示为白色(#FFF)。

(2) 含gazette类的段落元素显示为绿色(#0C0)。

(3) 其余所有元素显示为暗红色(#3000)。

文档中所有段落都按照最初在p规则中声明的属性值进行显示：字体大小为1em，行高1.6em，内边距10px。但是浏览器的默认边距0.9em仍然起作用，因为作者的CSS并没有对它进行重写。

1.4.3 根据顺序排序

假定一个作者样式表声明了两个规则，一个接着一个。这就对同一元素声明了多重规则。第二个规则的颜色将获胜(因此被采用)。在这种情况，颜色000被选择，这是黑色的十六进制表示。

```
p { color: #C00; }
p { color: #000; }
```

当多重规则具有相同的优先级、权重和来源时，规则来源总是在竞争中获胜。根据该规则，所有段落将被显示为黑色。当然可以改变权重使第一个声明被选择使用：

```
p { color: #C00 !important; }
p { color: #000; }
```

这些规则不再是平等的，因为作者的“!important”规则比正常的作者规则权重要高，因此所有段落将被显示为红色。

1.5　把理论应用于实践

迄今为止我们只详细介绍了CSS规则(而且我确信这引起了您足够重视)。而要把标准应用到实际的工作实践则完全是另一回事。为此我们将考察现代Web设计人员工具集里面的两个重要工具——它们或其中之一的魔力并没有引起人们的注意。

1.5.1　基于可靠浏览器进行构建

如果要构建一个站点并在性能很差的浏览器中测试，构建的代码就依赖于这种差的浏览器进行呈现。这就好像以沙地为地基建造房屋。一旦在其他浏览器或平台测试时，工作上的缺陷就会显现出来。因此应该在大家都认为标准兼容性比较好的现代浏览器上进行。在本章其余部分将看到，我们可以在代码中编写一些CSS hack(针对不同的浏览器写不同的CSS代码的过程就叫CSS hack)，以消除一些较小的问题。

这既不是冒充浏览器内行所做的辩解，也不是试图让您改换偏爱的浏览器。更确切地说，这种方案在构建站点时更节省时间和资源。如果站点一开始就是基于有缺陷的浏览器而构建的，则在与标准更兼容的浏览器上进行测试时将花更多的时间进行调试。在撰写本书时，有三种可供选择的浏览器：Opera、Safari或基于Gecko的浏览器(如Camino、Mozilla或Firefox)。

您可能注意到IE并没有出现在该列表中，很遗憾这不是故意的。虽然最近几年Windows的IE在标准实现上有很大提高，但与其他现代浏览器相比，在对CSS和XHTML等标准的支持上IE落在了后面。

勿庸置疑，我们仍不知道用户在什么地方点名需要更好的Firefox支持或改良的Opera布局。虽然每一个浏览器都认为自己是最好的，但要抓住顾客的心、占据一定的市场份额，还有一些工作要做。

1.5.2 理性对待hack

当然，在与基于CSS的布局协作时一定会有问题出现。虽然在过去几年里浏览器实现有很大改进，但仍不完美。除非在一个平台只支持一个浏览器，否则在不同浏览器/平台组合进行测试时肯定会遇到错误。表格布局技术的支持者可能会说，作为一种可行的布局方法，这些问题在层叠样式表中同样存在。然而这是浏览器的问题，并不是CSS规范本身的问题。

但是当每个浏览器都有自己的问题时，CSS就处于相当有利的位置。大多数问题(及其原因)都记录在文档中，并且在很多情况下都得到了圆满解决。在最广为流传的浏览器错误中，下面的示例就是其中之一。它不会是您遇到的唯一示例，但在对您有用的workaround(多个hack组成的迂回解决方案)中，它是其中一个非常好的示例。除非宕机、浏览器停止工作，几乎总是有解决的办法。

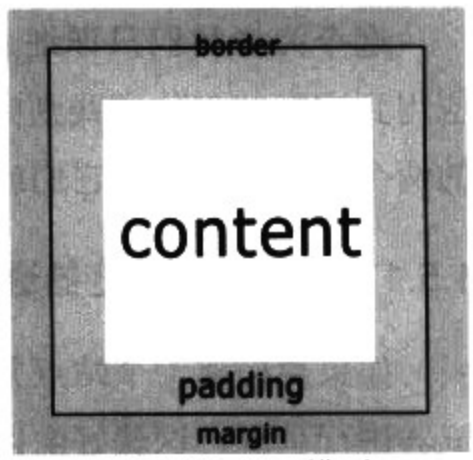


图1-18 box模型

1. 一个错误

根据CSS规范(http://www.w3.org/TR/CSS21/box.html)，文档树中的元素都有一个内容区，这可能是文本、图像或其他对象。另外，内边界、边框和外边界区域都可能环绕内容区域，如图1-18所示。

说明：

如果在华丽的黑白叶节点中看到box模型，您可能会搔搔头，Web设计师Jon Hicks已经构建了一个全色的三维图表，该图表值得我们检验一下。

现在把这三个“额外”区域(内边界、边框和外边界)的尺寸加入内容区的整个计算宽度和高度中。我们看一看实际的样式规则：

```
p#hack {
 border: 20px solid #C00;
 padding: 30px;
```

```
    width: 400px;
}
```

宽度属性声明了段落内的内容不超过400px。在此之上，box的每边(上、右、下、左)都有30px的内边距和20px厚的边框。如果要计算出段落的全部宽度，需要从左向右把box的属性加到最后的宽度中：

```
Left Border:              20
Left Padding:            +30
Content:                +400
Right Padding:           +30
Right Border:            +20
TOTAL WIDTH:           = 500 PIXELS
```

简而言之，内边距和边框在内容区域定义的宽度之外，这是规范要求的。

然而，IE/Win的老版本所实现的box模型是不正确的(更准确地说是“不标准的”)，它们把边框和内边距都包含在声明的宽度中。IE6是第一个计算正确的IE版本，而且只是在兼容模式下——也就是说要在标记的顶部有一个DOCTYPE声明。这些浏览器会错误地对待所定义的400px，认为它包含box的所有属性——内容宽度、内边距和边框。因此这些浏览器是这样计算的：

```
Declared Width:           400
Left Border:              -20
Left Padding:             -30
Right Padding:            -30
Right Border:             -20
CONTENT WIDTH:          = 300 PIXELS
```

当您试图保证针对所有浏览器是一致的设计时，即使是1px的差别也是不可接受的——一百多像素的差别足以使您使用令人信赖的表格。值得庆幸的是，这个问题是可以解决的。

说明：

值得注意的是，这个错误只发生在一个元素被定义了宽度，且定义了内边距或边框，或两者都被定义了的情况。避免这个问题的另一个策略是把内边距应用于某元素的父母并把宽度留给孩子——或反之。了解了这个呈现错误的原因有时比知道hack或修改方法更重要，可以帮助我们从策略上策划样式表的体系结构。

2. 解决方案

CSS hack为浏览器的不一致性(比如上面的IE 错误)提供了一个workaround，保证布局对所有浏览器看起来都是一样的。典型做法是，这些hack发现浏览器在CSS实现中的一个解析错误，允许对该浏览器隐藏或显示作者的CSS。实际效果是，允许对CSS兼容更好的浏览器提供“正确”值，而把“不正确”值发送给“数学技巧”差的浏览器。

为了解决这个小小的IE错误，可以求助于一些hack来保证在所有目标浏览器上的显示效果看起来是一样的：

```
p#hack {
 border: 20px solid #C00;
 padding: 30px;
 width: 400px;
}

* html p#hack {
 width: 500px;
 w\idth: 400px;
}
```

现在把一个CSS规则拆为两个。第一个规则包括应用于段落的边框和内边距信息，还包括期望的宽度400px。

第二个规则(以通配选择符*开头)包含一个技巧。如果用文字来解读* html p#hack规则，则为“选择id属性为‘hack’的所有p元素，这些p元素是某个html元素的后代，而这个html元素本身又是任一元素的后代”。要强调的是该规则的最后一部分，因为这里有一个hack。由于html是HTML和XHTML文档的根元素，它不能是任何其他元素的后代。既然第二条规则不匹配任何元素，为什么还要包含它呢？

实际上，在所有的IE版本(Windows和Macintosh，以后我们把Internet Explorer for Windows简写为IE/Win，Internet Explorer for Macintosh简写为IE/Mac)中，这条规则都会返回一个有效匹配，它错误地忽略了通配选择符而把这条规则解释为html p#hack。因此这条规则只对IE可见而被其他浏览器忽略。第一个属性声明了一个“不正确”的宽度500px，以保证在带错误的浏览器上为内容留下足够的空间。因为这些浏览器把内边距和边框包含在声明的宽度中，因此给它们的像素宽度必须与浏览器对规范的正确解释相匹配。由于html p#hack的权重比前一个规则更高，新的宽度值就重写了前面的值400px。

但是还没有完全解决问题，因为还有一个hack要完成。IE 6/Win和IE 5.x/Mac正确实现了这个box模型，因此不能把修改后的500px传给这两类浏览器。作为补救措施，第二个

宽度属性包含正确值400px。然而通过反斜杠把“i”转义(w\idth)，可以发现老版本IE的错误，从而对它们隐藏了这条规则。这种在hack中包含hack的方案修补了这个错误。

说明：

如果您感到不解，我能体会到您的痛苦。令人欣慰的是，有一些很好的资源可以帮助理解这个hack和其他CSS hack。建议您仔细阅读CSS讨论站点(下一节将详细介绍)，那里将深入介绍在这里用到的box模型hack以及其他方案(http://css-discuss.incutio.com/?page=BoxModelHack)。在那里几乎列出了所有其他样式列表hack，值得您细读(http://css-discuss.incutio.com/?page=CssHack)，有关避免不必要hack的技巧也值得您细读(http://css-discuss.incutio.com/?page=AvoidingHacks)。

3. 道路还很漫长

可以看到，由于存在大量特殊浏览器，使得CSS测试成为站点开发周期中必不可少的部分。对标准支持很好的浏览器(特别是与几年前的相比)，在自己代码中必定会遇到一些这样的浏览器错误。但是不必担心，这在站点开发过程中是很正常的。

每个站点设计人员都会在某些点遇到障碍。如果有人告诉您，他们在站点开发中没有遇到一个障碍，不要相信他们话。当您在调试自己的代码并不能解决这些问题时，可以(也应该)参考一些很优秀的站点。

如果在布局中遇到一个说不清的错误，知道一些可用的资源常常比马上就知道解决方法更重要。不过是为按期完成工程提供了一些保证。您所面临的问题很可能别人曾经遇到过。

CSS讨论

CSS讨论邮件列表(http://www.css-discuss.org)创建于1992年初，目前由Eric Meyer管理，他是CSS大师和早期Netscape标准的制定者。根据该站点的声明，邮件列表打算办成作者讨论CSS实际应用的地方。因此该列表办得非常成功，一批有志于CSS Web的优秀设计人员和开发人员都在列表中。通过这个CSS讨论，他们都愿意并热心于帮助其他Web专业人员，使他们能更好地分享协同工作的乐趣。

与列表本身一样有价值的是CSS讨论Wiki(http://css-discuss.incutio.com)，这是一个作者和编辑团体的站点，这个站点包含字体大小(http://css-discuss.incutio.com/?page=FontSize)、布局策略(http://css-discuss.incutio.com/?page=CssEditors)，当然还包

含CSS hack(http://cssdiscuss.incutio.com/?page=CssHacks)等有关信息。Wiki是一个值得仔细研读的站点，也是一个值得为之做出贡献的站点。

PIE站点

PIE(Position Is Everything)(http://www.positioniseverything.net)是一个详尽的CSS资源站点，当遇到浏览器难题时就可以参考该站点。PIE站点由John Gallant和Holly Bergevin维护，他们都是非常能干的CSS开发人员。PIE包含非常多的浏览器怪问题、workaround和未解决的错误——所有这些都是以清楚、易于理解的风格描述的，John和Holly因此而名声大振。

1.5.3 与hack有关的问题

当然，把hack直接写入CSS是完全可以接受的：在IE 5/Mac中找到一个错误、隔离有问题的规则、添加一个hack，然后转入下一个问题。这是大多数样式表开发人员采用的方案，被称为“随心所欲”方案。这种方案在处理浏览器一致性问题时非常灵活。此外，直接把workaround写入代码也相当安全：写hack、测试、下一个。

实际上，虽然这种方案非常有效，但这种即兴的hack管理方案在代码的整个生命周期内会造成问题。首先，由于有太多的hack，这些代码的重要程度变得很低，如：

```
#album-thumbs {
 float: left;
 list-style-image: none;
 list-style: none;
 margin: 0;
 padding: 0;
}

/* hide from MacIE5 \*/
* html #album-thumbs {
 display: inline;
 height: 1%;
 width: auto !important;
 width /**/: 90%;
}
/* hide from MacIE5 */

#album-thumbs a {
 display: block;
```

```
 float: left;
 padding: 6px;
 margin: 5px;
 width: 70px;
}

#album-thumbs a {
 \width: 60px;
 \width: 50px;
}
```

这是不是像在胡言乱语？必须注意，这位样式表作者在这个样式规则大杂烩中几乎没有什么注释。第一个注释中的反斜杠(/* ... */)是为了使IE/Mac停止解析CSS，直到遇到一个正确格式的注释(/* ... */)，因此不能算严格意义的注释。只有IE 5.x会解读\width声明，因此修复了CSS错误实现带来的问题。

天天看到这样的代码是什么感觉？如果您要读懂这段代码都很难，那么对于维护人员的困难将有多大！对团队成员、同事、实习生、对CSS一无所知的人——负责编辑CSS的任何人，这不是可行的hack管理方案。但是，如果暂时撇开代码的可读性(或根本就不可读)，这些规则是有效的CSS——修补不同浏览器/平台组合中的错误，而且还完成得很好。那么问题出在哪呢？

我们来想像一下CSS中充满这样杂乱代码的情形。需要在老式浏览器中呈现某个hack时会发生什么？或许一个有问题的浏览器忽略了升级期限，或者新版的IE在遇到针对Opera浏览器的hack时会停止解读这种样式表。有时可能需要编辑CSS并删除引进的某些hack。但如果这些hack传播行数不是30而是超过3000行呢？那又该如何呢？

1.5.4　编写hack的技巧

与其把样式表和与浏览器相关的hack(使CSS能在兼容性较差的浏览器中发挥作用)混在一起，不如把这些workaround放在与浏览器相关的样式表中，即把CSS与hack分离。虽然这种“hack隔离”严格地讲不是必须的，但这样做的好处是能保持CSS的整洁，请看图1-19。

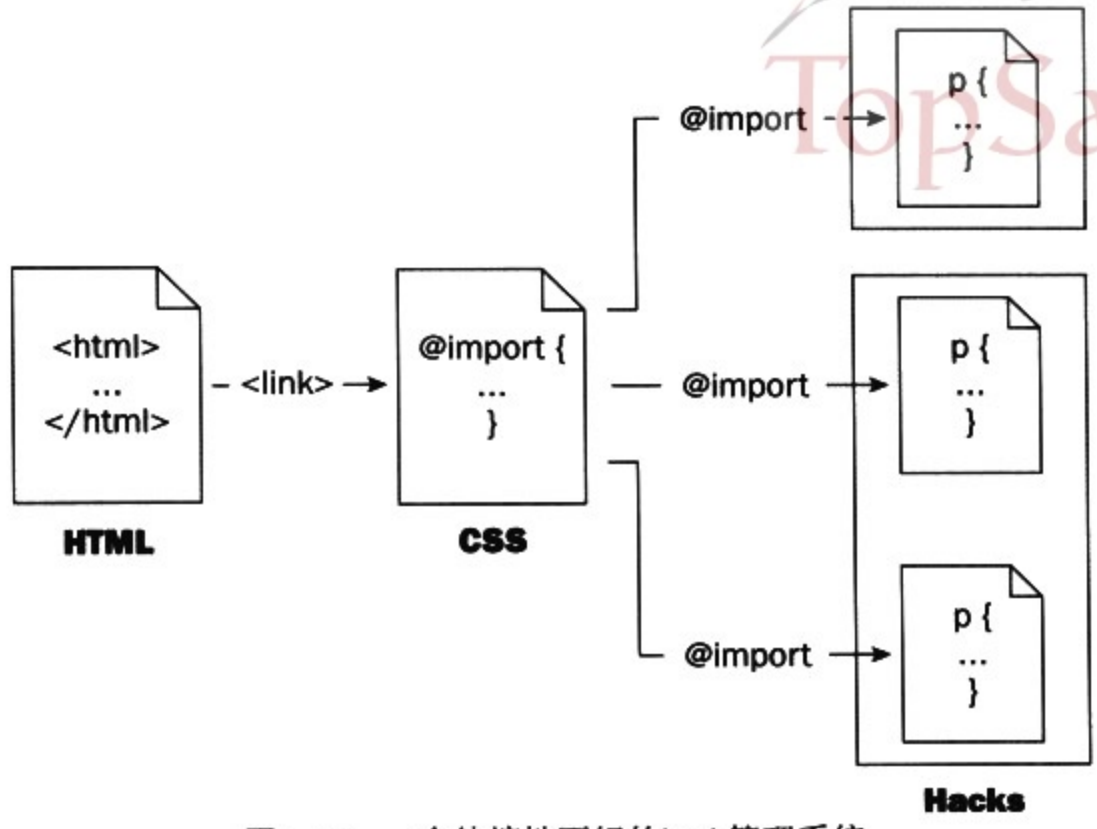


图1-19 一个伸缩性更好的hack管理系统

在这个模型中，有一个到样式表的链接——这里没有什么不寻常之处。但是第一个样式表仅仅起到网关的作用，通过它引入其他样式表，这些样式表通过@import规则引入。这是第一个引入的样式表，它包含站点的表现，是一个无hack的干净样式表。之后就可以简单引入与浏览器有关的样式表，这些样式表包含必要的hack以保证站点在所有目标浏览器上显示的一致性。

说明：

当然还有管理hack的其他方法。Web设计人员Mark Pilgrim介绍了他是如何在Web服务器上用用户代理检测来传送与浏览器有关的hack的(http://diveintomark.org/archives/2003/01/16/the_one_ive_never_tried)。这种方法被称为服务器端的内容协商。虽然这种方法需要Apache Web Server的相关知识并要求能对Apache Web Server及其mod_rewrite组件进行配置，但确实是非常好的方法。所以在技术水平比这里提供的解决方案要求还高时，这种服务器端的解决方案是一个非常好的选择。

在CSS文件网关的第一行将调用核心的样式表：

```
@import url("core.css");
```

这里没有什么令人惊异的地方。一个简单的@import规则就导入了core.css，这个文件

已在许多对CSS有很好支持的浏览器上经过了充分测试。现在只要用@import包含其他文件，就没什么可以阻止所有浏览器对它们进行解析的了。要做的是如何使这些CSS文件可选，使只有存在问题的用户代理消化这些规则。如果可能，还应该使“好”的浏览器意识不到这些hack。您会问，如何才能做到这一点？对了，解铃还需系铃人。我们可以用CSS hack编写一些更智能的hack。

首先，我们来处理IE 5.x/Win。该浏览器实现的CSS box模型是声名狼籍的，它所实现的box模型一定会使您的布局出现一些问题。因此，与其在核心样式表中添加workaround，不如利用Mid Pass Filter(MPF)，这是一个非常精妙的技术。

```
/* Import IE5x/Win hacks */
@media tty {
 i{content:"\";/*" "*/}} @import 'hacks.win.ie5.css'; /*";}
}/* */
```

MPF由CSS先驱和Web开发人员Tantek Çelik开发，它把一个样式表传递给IE 5.x/Win，并且只传给这种浏览器。所有其他用户代理将忽略这条规则，因为它们不会遇到与IE 5.x/Win一样的解析错误。

虽然IE 5.x/Mac对CSS的支持远比Windows平台的IE要好，但并不是没有错误。这种浏览器的开发截止于2003年中期，我们要自己处理偶发的布局问题。令人欣喜的是，有另一个筛选器可以处理这些问题，即IE 5/Mac Band Pass Filter。

```
/* Import IE5/Mac hacks */
/*\*//*/
@import url("hacks.mac.ie5.css");
/**/
```

IE 5/Mac Band Pass Filter也是由Tantek Çelik开发并由页面设计人员Doug Bowman推广，您已经猜到，这种筛选器只把CSS导入到IE5/Mac。其效果仍然与MPF相同。其他浏览器将忽略专为这种浏览器的怪问题而设计的样式表。

说明：

有关其他CSS筛选器和hack管理技术的介绍，请参考Molly Holzschlag的文章*Integrated Web Design: Strategies for Long-Term CSS Hack Management*(http://www.informit.com/articles/article.asp?p=170511)。这部分内容主要来自Molly的这篇优秀论文，在此未提及论文中的其他筛选器。

您可能会问自己，为什么这些解析错误是管理其他hack的最好办法？老实说，我只能泛泛地谈一下。把与浏览器有关的hack独立出来放在一个CSS文件中，以及保持核心样式表的相对纯洁，可以获得两个切实的好处。首先，这些workaround比较容易被其他人定位、维护和修改。第二，可以解决hack荒废的问题。如果您现在不需要支持某个浏览器，只需在核心样式表中删除几行就行了。

1.6 小结

在接下来几章中，我们将学习一些实际的专业策略，利用这些策略我们可以得到自己的样式表。本章介绍了一些非常新颖的XHTML/CSS设计基础、思想和目标，这些在以后各章中都将被用到。本章首先介绍了关于有效标记和良构标记的基础知识，然后介绍如何用CSS对页面的表现形式进行布局。这种方法缩减了页面大小，降低了将来重新设计的维护成本，增加了站点的可访问性。

第2章将深入介绍Google，这是本书所介绍几个公司中的第一个。由于有这些好处，他们对Blogger站点进行了重新设计，这就迫使我们对该案例进行研究，也确实说明该追求以CSS驱动的设计了，这是一个令人激动的时刻。

第2章

Google的blogger.com：翻转器和设计思想

1999年8月，一个称为Pyra Labs的小公司发布了一个新产品：Blogger to the Web。它不仅使该公司赢得了声誉和财富，而且也带来了一场有关博客(Blogger)的革命。

Blogger使人们(如您、您的朋友、实际上是任何一个人)可以发布一个Web站点，或更确切地说是博客。这个过程非常简单、快速且非常友好。而且Blogger还是免费的，它在该Web上的盈利很少，因为没有人会为网上的任何东西付费。

2003年2月，Google公司(您或许听说这个公司)收购了Pyra Labs，把Blogger并入Google旗下。随合同和资金而来的一些好事远非Blogger团队所能想像：在Google Toolbar上出现了一个“BlogThis！”按钮。每天大量的人使用Google ToolBar，这使blogger.com的访问量骤增。注册数似乎应该到了极限。但并没有，这是什么原因？

后来通过电话咨询，用户体验专家Adaptive Path对此也进行了分析。Stopdesign的设计师和CSS大师Douglas Bowman也到场进行分析。通过对Blogger.com访问者的行为进行分析，发现是站点设计本身出了问题：把新的访问者转换为新客户的基础部件停止了工作。

通过思想交流和计划的制定，经过六个月的开发，最后发布了一个新的设计。Blogger.com以新面孔出现并带来大量的新客户。从此以后，blogger.com和其他提供商占有了Internet通信的大部分份额。他们的客户数量达到了前所未有的高度。

在Bowman对blogger.com的重新设计中涉及大量巧妙的、也是有效的设计思想。本章介绍其中的一些设计思想，研究如何用当今最简洁的XHTML和最智能的CSS来重构这些设计。本章还涉及与IE有关的问题，并对问题浏览器提供一些迂回解决方案(如果可能)。

本章对一些想法进行了剖析，这些想法初看是不可能的，而实际上是有可能的，最初我们可能问的问题是“这在IE中有效吗”，但更合适的问题是“最简洁、最有远见的构造方法是什么”。

要把这里提供的解决方案完全强加到目前的开发过程可能不一定合适(这完全取决于开发人员)，但它们提供了整个开发过程的一个起点。从一个理想的解决方案入手，然后解决实际工作中遇到的困难，直到获得一个切实可行的解决方案。对许多开发人员来讲，这两点可能就是一点(是一样的)。在这种情况下，他们可以把这里介绍的技术用于以后构建的站点中。对于其他开发人员，这两点是有差距的，他们更偏重于保证设计在IE中百分之百是有效的，然后才考虑解决方案是否整洁。

因此，先阅读本章，理解课程内容，然后根据具体的工程和Web站点决定与理想解决方案的折中程度。

2.1 设计人员访谈

Douglas Bowman是一个有影响力的设计师，他对Blogger、Wired News和Adaptive Path站点进行了重新设计并公开发布，这些站点的重新设计非常成功。这把他推到与标准兼容的Web设计的最前沿。在Blogger工程工作一段时间后，Douglas Bowman接受邀请加盟Google，就任视觉设计总监，这是Google专门为他新设立的一个职位。

Q：首先要问的是，您对事情的进展感到满意吗？

A：我对整个工程的结果感到非常满意。对我来讲，一个工程成功的首要标志是能否满足或超过客户的目标。通常客户满意我才会满意。既然是这样，工程目标之一就是增加注册用户数。另一个目标则是增加Blogger的日常使用量(基于已有的用户)。我不能说出具体的数量，但我知道最后的结果远远超过Google的预期。

该工程包含多个方面，所有这些方面促使了Blogger重新设计的最后成功。Adaptive Path和Stopdesign与Google一起对Blogger的主页和注册系统进行了重新设计和简化。Stopdesign与五个设计人员签订了合同，以帮助创建三十多个新的用户模板。为了增加影响，Google努力扩展Blogger的特色和能力，使之位于同行前列。用户配置文件、注释、驻留在BlogSpot的新博客主页(这是一个没有广告的博客主页)和博客搜索只是新特色的几个方面，这些特色是在重新设计的同时或前后添加的。

Q：哪些设计是团队感到最满意的？哪些又是站点访问者感到最满意的？

A：我认为这可能取决于您问的是团队中的哪一个人。我们先从Google的开发人员和工程人员开始，我认为他们最满意的是设计系统，以及页面的极易扩展性和改变性。设计非常简洁和直观。显然，它所采用的是非常简洁的HTML，布局全部由CSS控制。Google最终采用了我们为主页和注册页面提供的XHTML模板和CSS，并把它们作为重新设计整个应用程序的基础。除了Google所需的页面类型外，我们还提供了一些通用模板，用这些模板能很快构成站点新的部分。

如果您问的是产品管理人员，他们会说他们最喜欢这个简洁的设计。特别是关于新主页的交流方式很方便(也就是博客)，以及用Blogger马上就可以开始创建一个新博客。

Blogger的访问者可能没有更多留意站点的设计。事实上，如果我们的工作做得好，用户可能对Blogger的外观稍微有所关注，但他们不会对设计太关注。他们能立即感受到好处，并在尽可能短的时间内找到发布自己博客的路径。用户马上就发现，在建立自己的博客时，有更多的模板可选择。在定制博客以表达自己的声音和个性时，有非常丰富的预制品可供选择。

Q：在本章后面的内容里，我们将仔细研究有关Blogger.com圆角的代码，但这与您在站点中所用的解决方案不同。我们的方法不需要额外的div，但会牺牲一些跨浏览器方面的性能，因为IE接收的是无角样式。如果再次开发该项目，您会考虑采用这样的方案吗？或者，作为一个设计人员，您会要求设计中的每一个元素都遵守规则吗？

A：如果完全由我来选择并且我也是唯一一个与代码打交道的人，我会选择更瘦的、没有额外div的方案。我的优势在于能准确了解利益的平衡点所在，能准确知道更高级的CSS是如何工作的。更瘦、更简洁的HTML总是好的，特别是在HTML可删除的情况下更是如此，这些可删除的HTML是为了进行样式控制而特意插入的。页面模块代码不仅应该更简洁，而应该更稳定、更少依赖分类div的准确数目。

在最近几个小型工程中我采用了多种方案：只给IE一个样式基本集，而给其他CSS能力更强的浏览器则增加一些高级样式。这个方案的术语是“渐进增强(Progressive Enhancement)”，这个术语是Steven Champeon在他几年前为Webmonkey所写的文章中提出的。对某些浏览器，可以提供一个基础设计，但运作得很好；对于能力更强的浏览器，则在此设计基础上添加更多的高级设计。

这个案例中，对于整个站点设计中的圆角，其主要作用是给Blogger一个简洁友好的用户体验。大量(可能是大多数)Blogger用户仍把IE作为默认浏览器。如果

我们采用渐进增强方案，这些IE用户看到的页面设计就与新的Blogger“外观”不完全一致。

对于其他工程，使所有设计细节在IE中都正确并不重要。但对于Blogger重新设计，Google的目标是为了满足更多用户，对他们的技术要求也更少——因为仍有大量用户还在使用IE浏览器，因此为简洁化HTML而对IE浏览器的美学设计做一定让步，这种方案未必是一个好的方案。

Q：重新设计给Blogger带来了哪些好处？是财政上的还是仅仅吸引了大量客户？

A：前面已提到，我对具体的数字不能确定。但我可以说，在主页和注册系统重新设计后，Blogger的注册用户数急剧增加。修改后的主页使更多用户进入注册系统。更简洁的注册系统与以前相比，拥有更少的页面。该设计全程指导用户尽可能快地、尽可能容易地创建和发布一个博客，整个过程不超过五分钟。这些变化基本保证了成功。但即使Google也不知道究竟有多成功。

Q：回过头来看，所有工作您都是以不同方式完成的吗？

A：对于我所完成的每个大的设计，我常常发现有更好的方法，并且需要在一到两周内完成重新设计。因此会加班加点对设计进行调整，使编码变得更容易。一个以前从没用过的CSS选择符突然变得有意义了，并且我知道了它的各种用法。

特别是在Blogger的CSS中，用了很多我认为是不必要的float属性。一些是专门用于修补浏览器错误的。其余是用于更合法的目的，例如用于包含其他嵌套的悬浮元素。

我也希望我能更多地涉及该工程的其他方面。Google的团队对我们提供的设计和模板进行了大量扩展。但我希望我们能帮助或监督某些扩展，以保持设计的一致性，保证设计方案的质量。但预算经常是一个限制因素，仅仅是因为预算不够而无法使Stopdesign 或Adaptive Path涉及工程每个大的阶段。

Q：Blogger重新设计给您提供了多大好处？其他大公司看到过最后完成的产品吗？它们打算走相同的道路吗？

A：Blogger重新设计是一个大工程。与Adaptive Path的合作始终是一个有趣的学习经历；与Google团队的合作产生了很好的结果，因为与他们的合作非常自然。如果Google有机会成功开展一个大的工程，将更多关注Adaptive Path和Stopdesign。由于人们确实注意到了Blogger的重新设计，这两个公司开展了几个新工程，并得到Google专业人士的帮助。

我认为Blogger的重新设计不过是基于其他设计基础之上的一个更完整的范

例，它使大公司相信现在基于标准的设计是唯一出路。标准的所有优势与两个设计顾问的能力和天赋相得益彰。而更重要的是，该工程是为高端客户开发的，其产品天天被大量的人使用和回复。勿庸置疑，Blogger的重新设计已经引起了大量关注，并将引起更多的关注。它是体现有效设计及其合理实现重要性的突出范例。我们希望Blogger能支持其他开发人员的案例，如果设计人员请求对已正确实施的设计进行评估，他能给出更充足的理由。

Q：Blogger三个竞争对手中有两个从现在开始已使用基于标准的设计。您认为这是Blogger采用标准的主要驱动力还是开发站点的必然趋势？

A：我认为Google没有必要这样想：“噢，我们的竞争对手都采用基于标准的设计，我们也要紧跟趋势而采用相同的设计思想”。Google在邀请我们时已有几个简单目标，可能只是没有想出如何实现这些目标而已。Google知道我们能帮助他们解决问题并极大地改进用户体验。Google和Adaptive Path也知道StopDesign在实现它的设计时不乏智慧，因此我认为基于标准的解决方案是默认的方案。

有时，客户能隐约意识到标准的好处。但是一旦他们开始用一个基于标准的解决方案来建立自己的站点时，他们才能真正了解并为之激动不已。现在这种好处还仅停留在字面上，一旦自己实践了才能真正体会到。

无论用什么CMS(Content Management Systems，内容管理系统)或脚本语言输出代码，越简洁越瘦的HTML总是更好的。这不仅使工作更容易、更快速，而且常常是立即就可进行解析，不需要剖析多重表、列和单元。一旦实现了基本设计且相当稳定后，Google的工程人员和开发人员就可继续在已有代码基础上做一些改变，同时我们驻留CSS文件并继续对设计做一些小的改变。这种两边同时迭代的方法被证明是一种方便的方法。

2.2　CSS驱动的翻转器

如果说CSS使Web变得简洁，翻转器就是一个很好的示例。翻转就是用户移动鼠标掠过页面某个部分时，该部分中的一个图像(或颜色)变换为另一个图像(或颜色)的动作。大约在2001年以前，要实现这种效果，唯一可靠方法是使用JavaScript，并编写类似如下代码。

```
<html>
 <head>
  <title></title>
<script type="text/javascript">
```

```
<!--
function SwapOut()
    {
    document.getElementById('picture').src = 'picture-rollover.jpg';
    return true;
    }

function SwapBack()
    {
    document.getElementById('picture').src = 'picture.jpg';
    return true;
    }
-->
</script>
 </head>
 <body>
   <p><a href="" onmouseover="SwapOut()" onmouseout="SwapBack()"><img
id="picture"src="picture.jpg" width="100" height="150" /></a></p>
 </body>
</html>
```

谢天谢地，要到达这个目的，CSS提供了大量方法，令人欣喜的是这些方法都非常简单。下面我们将介绍这些方法。

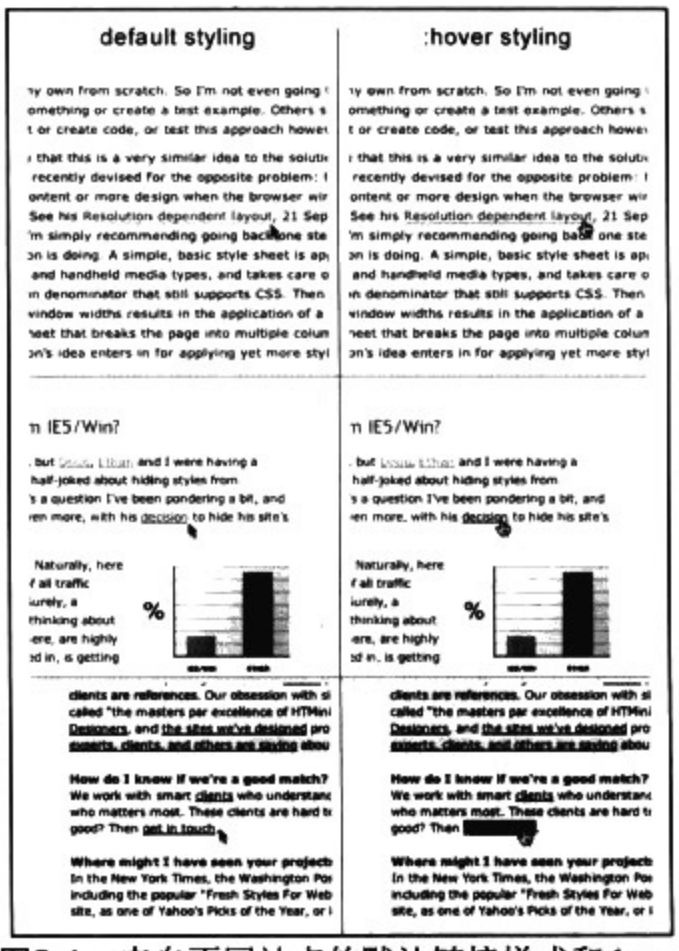


图2-1 来自不同站点的默认链接样式和:hover链接样式

2.2.1 改变链接的颜色和背景色(简单)

改变链接的颜色和背景色是所有CSS翻转器技术中最简单(也是最常用)的。它用于提醒用户鼠标位于一个超链接上。图2-1是关于翻转器的演示示例。

访问下列链接可得到有关翻转器的大量示例：

- http://pootato.org/examples/rollover.html
- http://pootato.org/tutorials/css/css-rollovers/

下面我们来研究前面的示例，看看如何将其用于自己的工作中。

1. XHTML

XHTML代码如下所示：

```
<p>If you're interested then <a href="">bung me an email</a> and we can
talk about what you want </p>
```

因此只需要一个简单的标签。那么CSS呢？

2. CSS

CSS代码如下所示：

```
a {
 border-bottom: 1px solid #eee;
 color: #d17e62;
 text-decoration: none;
}
a:visited {
 border-bottom: 1px solid #eee;
 color: #9d604c;
 text-decoration: none;
}
a:hover {
 background-color: #ffffda;
 border-bottom: 1px solid #ddd;
 color: #c30;
 text-decoration: none;
}
```

注意，这些规则的顺序很重要。第一条规则a{}影响所有链接；第二条规则a:visited{}影响用户已经访问过的链接(这由浏览器的缓存决定)；第三条规则a:hover{}影响当前鼠标停留在之上的链接。

按照CSS层叠逻辑(http://www.htmlhelp.com/reference/css/structure.html#cascade)，每条规则都比出现在它之前的优先级高。因此一个正常链接的样式将被一个已访问链接重写，已访问链接的样式在用户停留之上时被重写。确实很简单，但是令人惊异的是很多人在写这些规则时把顺序弄错了。

2.2.2 改变链接的颜色和背景色(复杂)

要实现快速、低带宽的翻转器需要非常高的技巧，这也是Bowman在Blogger的主页中所用到的技巧。其实也没什么，只不过是改变了一个元素的背景色而背景之上的图像(称

为内联图像或CSS背景图像)保持不变。

我们来看看演示，Blogger.com主页的四个链接如图2-2所示。

图2-3是把鼠标移动到其中一个链接之上时的情形。

图2-2 Blogger主页上的四个链接(其余部分已淡去以便能看清这些链接)

可以看到背景色已改变且单词thoughts已变为黑色。而实际按钮图像完全没变。

这是如何实现的呢？首先我们来看看组成这一部分的XHTML、CSS和图像。

1. XHTML

XHTML代码如下所示：

```
    <li id="wpub"><a href="https://www.blogger.com/tour_pub.
g"><strong>Publish</strong>
    thoughts</a></li>
      <li id="wcon"><a href="https://www.blogger.com/tour_con.
g"><strong>Get</strong>feedback</a></li>
    <li id="wshr"><a href="https://www.blogger.com/tour_shr.
```

```
g"><strong>Post</strong>photos</a></li>
    <li id="wpst"><a href="https://www.blogger.com/tour_pst.g"><strong>Go</
strong>mobile</a>
    </li>
```

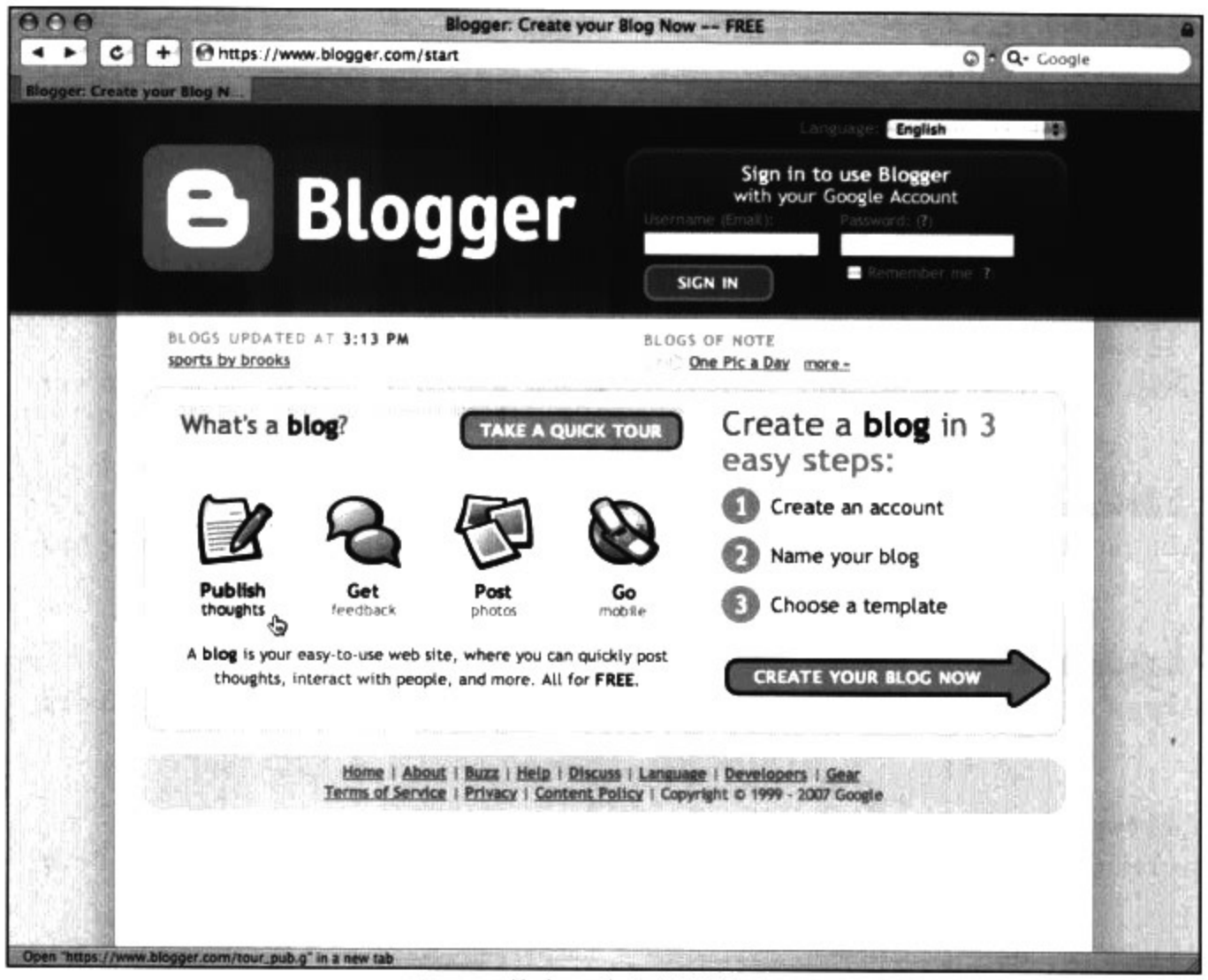


图2-3　其中一个翻转链接演示

2. CSS

CSS代码如下所示：

```
ul {
 list-style: none;
 margin: 0;
 padding: 0;
}

ul li {
 float: left;
```

```
 margin: 0;
 padding: 0;
}

ul li a {
 color: #777;
 display: block;
 padding: 80px 10px 5px;
 text-align: center;
 text-decoration: none;
 width: 75px;
}

ul li#wpub a {
 background: transparent url(icon_wpub.gif) no-repeat top center;
}

ul li#wcon a {
 background: transparent url(icon_wcon.gif) no-repeat top center;
}

ul li#wshr a {
 background: transparent url(icon_wshr.gif) no-repeat top center;
}

ul li#wpst a {
 background: transparent url(icon_wpst.gif) no-repeat top center;
}

ul li a strong {
 color: #000;
 font-size: larger;
}

ul li a strong {
 color: #000;
 display: block;
 font-size: larger;
}

ul li#wpub a:hover,
ul li#wcon a:hover,
ul li#wshr a:hover,
ul li#wpst a:hover {
```

```
 background-color: #f8f2eb;
}

ul li a:hover {
 color: #000;
}
```

3. 图像

要实现这种效果，所用图像必须有透明部分，使图像(或父元素)的背景色能显示出来。在所有情况下，图2-4中的图像都有透明部分。透明部分用格状图案表示，因此我们能看清哪些地方背景色是可以透过，哪些地方是不能透过的。

图2-4 格状图案指出透明区

2.2.3 所要完成的工作

最初的显示结果如图2-5所示。

现在我们一行一行地研究CSS，看看每一部分有什么效果。首先去掉每个列表项前的圆点，结果如图2-6所示。

```
ul {
 list-style: none;
}
```

- Publish thoughts
- Get feedback
- Post photos
- Go mobile

图2-5 无样式时的显示

Publish thoughts
Get feedback
Post photos
Go mobile

图2-6 删除圆点

然后删除无序列表的内、外边距，结果如图2-7所示。

```
ul {
 list-style: none;
 margin: 0;
 padding: 0;
}
```

Publish thoughts
Get feedback
Post photos
Go mobile

图2-7 删除内外边距

这样，当我们确定页面中已完成列表的位置时，就不必考虑无序列表内、外边距的默认浏览器设置。

下面为列表项添加样式。首先把每个列表项的float属性设置为left，使它们不再按列显示，而是一个挨一个地水平排列，如图2-8所示。同时还要去掉每个列表项的内、外边距。

```
ul li {
 float: left;
 margin: 0;
 padding: 0;
}
```

现在设置链接的样式。首先设置一个字体颜色(如图2-9所示)。

```
ul li a {
 color: #777;
}
```

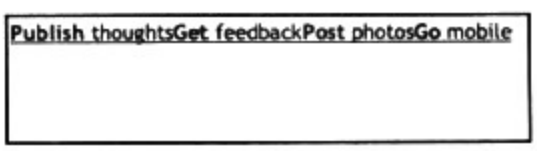


图2-8 把列表中的每一项水平排列

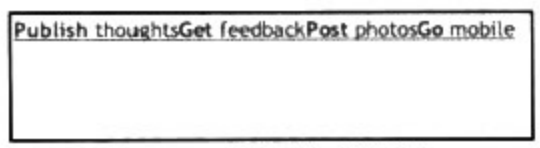


图2-9 设置字体颜色

接下来把链接的display属性设置为block，width属性设置为75px(等于将要用到的图像宽度)，如图2-10所示。

```
ul li a {
 color: #777;
 display: block;
 width: 75px;
}
```

现在必须插入一些用来放置图像(一会儿再添加图像)的空白区域，如图2-11所示。为此在顶部添加80px的内边距(75px放置图像，5px是图像与文本之间的间隙)。

```
ul li a {
 color: #777;
 display: block;
 padding: 80px 0 0 0;
 width: 75px;
}
```

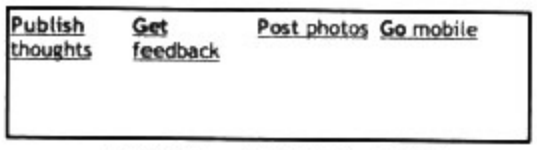


图2-10 按块显示元素

图2-11 插入空白以放置图像

然后在左边和右边添加10px的内边距，底部添加5px的内边距(如图2-12所示)。

```
ul li a {
color: #777;
display: block;
padding: 80px 10px 5px;
width: 75px;
}
```

为了完成通用链接样式设置，把链接文本设置为居中对齐并删除链接文本的下划线(如图2-13所示)。

```
ul li a {
 color: #777;
 display: block;
 padding: 80px 10px 5px;
 text-align: center;
 text-decoration: none;
 width: 75px;
}
```

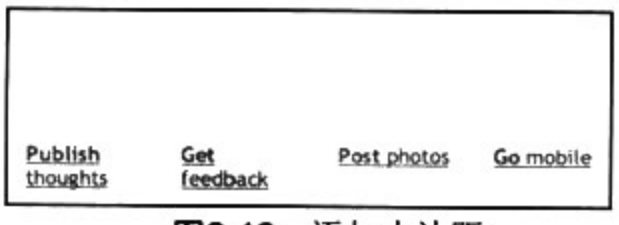


图2-12 添加内边距

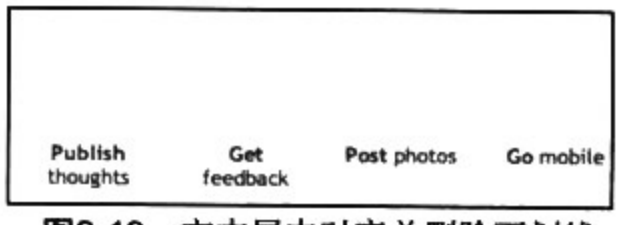


图2-13 文本居中对齐并删除下划线

现在该为每个链接添加背景图像了，如图2-14所示。

```
ul li#wpub a {
 background: transparent url(icon_wpub.gif) no-repeat top center;
}

ul li#wcon a {
 background: transparent url(icon_wcon.gif) no-repeat top center;
}
```

```
ul li#wshr a {
 background: transparent url(icon_wshr.gif) no-repeat top center;
}

ul li#wpst a {
 background: transparent url(icon_wpst.gif) no-repeat top center;
}
```

下面把每个链接的第一个单词变得更粗更大，如图2-15所示。

```
ul li a strong {
 color: #000;
 font-size: larger;
}
```

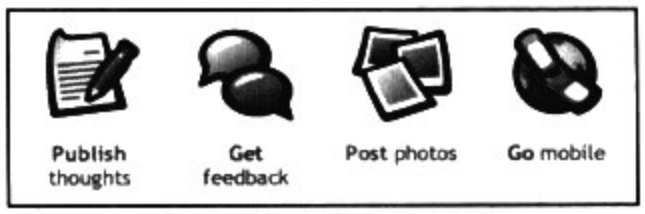


图2-14 添加背景图像

图2-15 把链接的第一个单词变得更粗更大

为了保证每个链接的第二个单词位于新行(如图2-16所示)，应添加一个规则。

```
ul li a strong {
 color: #000;
 display: block;
 font-size: larger;
}
```

把这些代码添加到CSS使翻转器能正常工作。

当用户把鼠标悬停在某个链接之上时，下列规则改变该链接的背景色(如图2-17所示)。

```
ul li#wpub a:hover,
ul li#wcon a:hover,
ul li#wshr a:hover,
ul li#wpst a:hover {
 background-color: #f8f2eb;
}
```

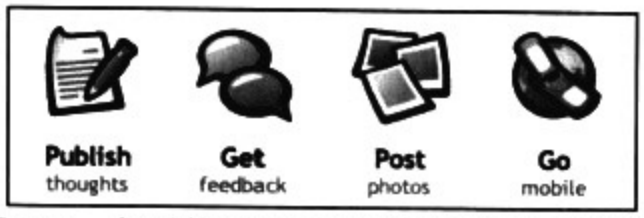


图2-16 把链接的第二个单词强制显示在新行

图2-17 改变背景色

最后要做的是添加一个小规则，它改变:hover链接样式的文本颜色，使之从灰色变为黑色，如图2-18所示。

```
ul li a:hover {
  color: #000;
}
```

一切都完成了！

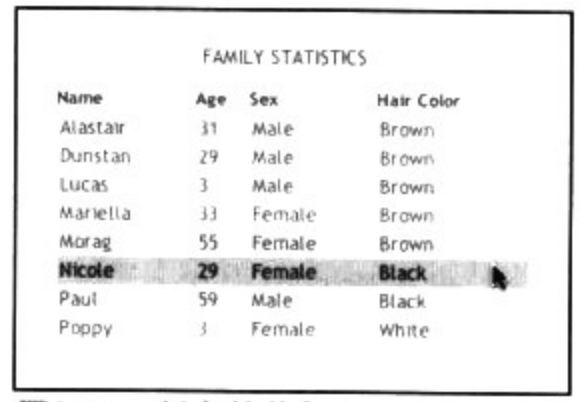
图2-18 最终效果演示

2.2.4 改变表格行的背景色

表格中交替行不同颜色的使用成为一种公认的设计思想(比如iTunes)，这给长表提供了一种新的结构并使人们浏览一行信息非常容易。

如果与一个悬停效果结合起来，使鼠标指针停留在某行之上时醒目显示该行，就会产生一个富有吸引力的功能外表，如图2-19所示。

悬停效果的代码会更复杂。按照前面的示例，需要用XHTML和CSS给表行添加样式。



FAMILY STATISTICS

Name	Age	Sex	Hair Color
Alastair	31	Male	Brown
Dunstan	29	Male	Brown
Lucas	3	Male	Brown
Mariella	33	Female	Brown
Morag	55	Female	Brown
Nicole	29	Female	Black
Paul	59	Male	Black
Poppy	3	Female	White

图2-19 行交替着色和醒目显示示例

1. XHTML

XHTML代码如下所示。

```
<table cellpadding="5" cellspacing="0" border="1">
<caption>Family Statistics</caption>

<thead>
 <tr>
  <th>Name</th>
  <th>Age</th>
  <th>Sex</th>
  <th>Hair Color</th>
</tr>
</thead>
```

```
<tbody>
 <tr class="odd">
  <td>Alastair</td>
  <td>31</td>
  <td>Male</td>
  <td>Brown</td>
</tr>
<tr class="even">
  <td>Dunstan</td>
  <td>29</td>
  <td>Male</td>
  <td>Brown</td>
</tr>
<tr class="odd">
  <td>Lucas</td>
  <td>3</td>
  <td>Male</td>
  <td>Brown</td>
</tr>
<tr class="even">
  <td>Mariella</td>
  <td>33</td>
  <td>Female</td>
  <td>Brown</td>
</tr>
<tr class="odd">
  <td>Morag</td>
  <td>55</td>
  <td>Female</td>
  <td>Brown</td>
</tr>
<tr class="even">
  <td>Nicole</td>
  <td>29</td>
  <td>Female</td>
  <td>Black</td>
</tr>
<tr class="odd">
  <td>Paul</td>
  <td>59</td>
  <td>Male</td>
  <td>Black</td>
</tr>
<tr class="even">
```

```
    <td>Poppy</td>
    <td>3</td>
    <td>Female</td>
    <td>White</td>
    </tr>
  </tbody>
</table>
```

2. CSS

要设置表格的样式，下面的大部分CSS都是必需的。实现悬停效果的规则已经用粗体字标出。

```
table {
 background-color: #fff;
 border: 1px solid #ddd;
 empty-cells: show;
 font-size: 90%;
 margin: 0 0 20px 0;
 padding: 4px;
 text-align: left;
 width: 300px;
}

table caption {
 color: #777;
 margin: 0 0 5px 0;
 padding: 0;
 text-align: center;
 text-transform: uppercase;
}

table thead th {
 border: 0;
 border-bottom: 1px solid #ddd;
 color: #777;
 font-size: 90%;
 padding: 3px 0;
 margin: 0 0 5px 0;
 text-align: left;
}

table tbody tr.odd {
 background-color: #f7f7f7;
```

```
}

table tbody tr.even {
 background-color: #fff;
}

table tbody tr:hover {
 background-color: #ffe08e;
}

table tbody td {
 color: #888;
 padding: 2px;
 border: 0;
}

table tbody tr:hover td {
 color: #444;
}
```

3. 所要完成的工作

我们不准备仔细研究每一行CSS代码，因为这与翻转器完全不相关。不过还是要仔细研究一下最后几个规则。

首先要注意的是，给两个行类(.odd和.even)分别设置一个背景色：

```
table tbody tr.odd {
 background-color: #f7f7f7;
}

table tbody tr.even {
 background-color: #fff;
}
```

这样就可创建行交替着色的效果。

然后设置一个规则，当鼠标悬停在某行之上时改变该行的背景色：

```
table tbody tr:hover {
 background-color: #ffe08e;
}
```

最后，当鼠标悬停在某行之上时，下列规则将改变这一行所含文本的颜色，使之更暗

一点以便突出新的背景色：

```
table tbody tr:hover td {
 color: #444;
}
```

真是既精彩又简单！

2.2.5 改变文本颜色

本节最后要研究的悬停应用是，当用户的鼠标掠过某个div或类似元素时，将醒目显示其文本(纯文本或链接文本)。

这有什么用呢？想像一下一个包含很多链接的站点。开发人员常常要给链接添加样式使之变为明亮的颜色，这样链接就可以从周围的文本中突出显示。如果每一页只有少许几个链接，这个做法当然很好。但如果站点每个页面包含上百个链接，又会怎么样呢？由于有数百个这样的链接，大量的明亮色彩将伤害用户的眼睛。如果所看到的是一片色彩的海洋，又怎样能看到设计的精妙之处呢？

解决方案(或解决方案之一)很简单。先把这些链接的明亮颜色隐藏起来，直到用户移动鼠标掠过该页的相关部分，这样用户第一眼看到站点就比较平和一些。只有在用户移动鼠标悬停在他/感兴趣的部分时，该链接才被突出显示。

这相当有效。我们看看这是如何重构的？

1. XHTML

XHTML代码如下所示。

```
<div id="links">
 <h3>Blogmarks</h3>

  <p>A collection of miscellaneous links that don't merit a main blog
posting, but which are interesting none-the-less.</p>

  <ul>
   <li><a href="">What WordPress is currently doing to combat comment
spam</a>.</li>
   <li><a href="">Mobile web tools</a>, from <a href="">Pukupi</a>.</li>
   <li><a href="">The photography of E.J. Peiker</a>.</li>
   <li><a href="">Some handy tips for advanced Google use</a>.</li>
   <li><a href="">Make your own church signs</a>, or <a href="">view some
```

```
real ones</a>.</li>
      <li><a href="">Michael Heilemann</a> is doing a great job with his new
<a href="">WordPress</a> theme, <a href="">Kubrick</a>.</li>
       <li>I'm late to the party, but <a href="">Dan</a> has a
<a href="">book out</a>.</li>
      <li><a href="">A crazy concept for laying our housing estates</a>.
</li>
      <li><a href="">Spiderman reviews crayons</a>.</li>
     <li>Some <a href="">beautiful images</a> from photographer
<a href="">Greg Downing</a>.</li>
      <li><a href="">Lots of links from the Link Bunnies</a>.</li>
      <li>Nice <a href="">&#8220;when I was a child&#8221;</a> sort of post
frin Stuart.</li>
      <li><a href="">How much does SafariSorter cost?</a></li>
      <li>Some handy <a href="">maintenance tips for Mac owners</a>.</li>
      <li><a href="">Ming Jung</a>, <a href="">Anil Dash</a>, and I get
    <a href="">interviewed for HBO's Real Sex</a>.</li>
     </ul>
    </div>
```

2. CSS

CSS代码如下所示。

```
div#links {
 color: #333;
 border: 2px solid #ddd;
 padding: 10px;
 width: 240px;
}

html > body div#links {
 width: 220px;
}

div#links ul {
 margin: 0 0 0 19px;
 padding: 0;
}

div#links ul li {
 list-style-image: url('list-dot.gif');
 margin: 0 0 .5em 0;
}
```

```
html > body div#links ul li a {
 color: #333;
 text-decoration: none;
}

div#links:hover ul li a {
 color: #0000ff;
 text-decoration: underline;
}

div#links ul li a:hover {
 background-color: #ffff66;
 color: #000;
 text-decoration:none;
}
```

3. 图像

我们用一个“小方点”图像取代浏览器默认的“列表圆点”，如图2-20所示。

图2-20　替代“列表圆点”的图像

4. 所要做的工作

我们仍然不会一行行地分析CSS，因为这与讨论的翻转器技术不相关。不过仍有一些地方值得仔细研究。

首先设置div的默认文本颜色。我们选择深灰色，因为灰色能给人平静安详的感觉，这正是我们努力要达到的目标。

```
div#links {
 color: #333;
}
```

然后设置链接的样式，使之与灰色文本匹配，删除使之突出于周围的所有东西。

```
html > body div#links ul li a {
 color: #333;
 text-decoration: none;
}
```

用子选择符>设置链接的样式，这样IE(IE不解析包含子选择符的规则)就不会采用该规则。这样做的原因请参见下一小节“5. 对IE的处理”。

现在分析两个:hover规则。当用户把鼠标移到div上时，第一个规则将被激活。这个规则使链接的颜色由灰色变为蓝色，并显示文本装饰下划线。

```
div#links:hover ul li a {
 color: #0000ff;
 text-decoration: underline;
}
```

当用户正好把鼠标移到链接之上时，第二个:hover规则将被激活。这个规则把文本颜色变为黑色、删除下划线，并把链接的背景色变为浅黄色。

```
div#links ul li a:hover {
 background-color: #ffff66;
 color: #000;
 text-decoration: none;
}
```

5. 对IE的处理

除链接元素之外，IE 6不理解其他元素的:hover选择符。因此，Firefox,、Opera、Safari和IE都清楚下列规则的含义：

```
a:hover {
 color: red;
}
```

但只有Firefox,、Opera和Safari清楚下列规则的含义：

```
div:hover {
 color: yellow;
}

div:hover a {
 color: green;
}
```

为什么会这样呢？这正是该示例要介绍的内容。想像一下您正在用Firefox浏览一个Web站点。加载页面，CSS把所有文本和链接都变为灰色，对您来说，一切都很完美。瞄准一个感兴趣的部分。把鼠标移到上面，链接立刻变为浅蓝色。现在可以看清哪个是链接，哪个是纯文本了。

现在想像一下用IE浏览相同的站点。加载页面，CSS把所有文本和链接都变为灰色。瞄准一个感兴趣的部分。把鼠标移到上面，结果什么也没发生。没有链接突出显示，留给您的是一片毫无反应的灰色文本。为什么会这样？这是因为IE不知道鼠标正悬停在一个div之上，因此它不能对这个链接使用一个更明亮的样式。

您明白了为什么在IE中一开始不能把链接立即变为灰色。如果浏览器不能在合适的时刻把链接的颜色变回蓝色，就没有办法把它们变为灰色。因此要采用子选择符。

对IE来讲，用子选择符隐藏样式时还要注意：IE/PC不理解子选择符，但是IE 5/Mac却能理解！因此，任何时候用“>”屏蔽IE时，必须记住屏蔽的只是IE/PC，不是IE/Mac。

2.2.6　改变链接的背景位置

基于CSS的翻转器还有第二种实现方案。这种方案不仅关注背景色的变换，也关注图像的变换。这要用到一个重要的方法：背景定位。

如果您从没听说过这种思想，不用担心，这非常容易，不过我们需要做一些额外的解释。

1. 利用嵌入图像构造翻转器

假设有如图2-21所示的页面布局，并且要求把其中的图片变得活泼一点——添加一个翻转器、或许还要加一点阴影、或其他更吸引人的东西。

您可以实现，对吗？好了，如果规定不能用XHTML，您还能实现吗？我们先看看下列代码：

```
<div id="sidebar">
 <div class="box">
  <h2>Lucas says...</h2>
 <div id="photo-lucas">
  <img src="lucas.jpg" width="150" height="100" alt="Portrait of small
boy" />
 </div>
```

```
    <p>Hello there, my name is Lucas. I am 3-years old and the bi-lingual
monkeychild of an English man and an Italian woman.</p>

     <p>I like cars and trucks and buses and trains and almost anything
that moves. I'm not so keen on sprouts or the Welsh.</p>

     <p>My Grandma has a dog, called Poppy. She's small and cute and she
widdles when she gets excited.</p>
    <p>When I grow up I want to be like my Uncle... what a guy he is...</p>
    </div>
   </div>
```

如果您很幸运地由于经常与XHTML打交道，因此对XHTML很熟悉，那么这就是小事一桩。其实利用神奇的Adobe Photoshop和一些CSS，就可以很快得到这样一个布局。这个布局不仅插入了一个新图像并对它进行了重新布局，而且当用户把鼠标移到Lucas says这个框内时，其功能相当于一个翻转器。

图2-22显示的是最后的结果。这个页面被显示了两次，以便能看清正常状态的页面(左边)和“翻转”状态的页面(右边)。

图2-21 能用CSS替换这张照片吗?

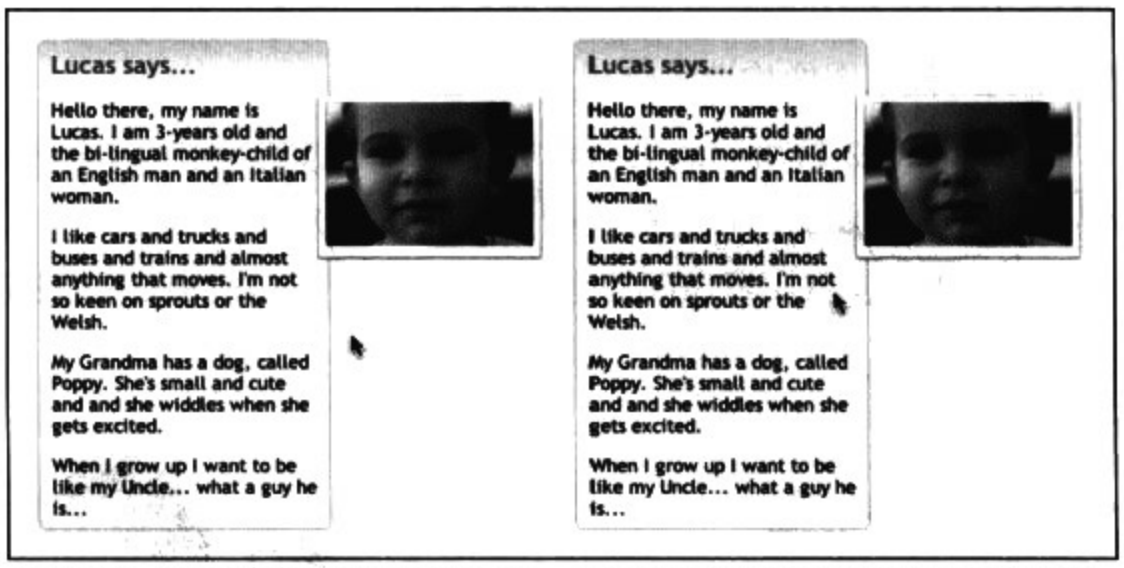


图2-22 最后得到的布局和翻转器，鼠标移到文本上时触发翻转器显示彩色图片

下面看看这是如何实现的。

2. XHTML

我们把注意力集中在照片部分的代码，这部分代码已经用粗体字标出。

```
<div id="sidebar">
 <div class="box">
  <h2>Lucas says...</h2>

  <div id="photo-lucas">
   <img src="lucas.jpg" width="150" height="100" alt="Portrait of small
boy" />
  </div>

  <p>Hello there, my name is Lucas. I am 3-years old and the bi-lingual
monkeychild of an English man and an Italian woman.</p>

  <p>I like cars and trucks and buses and trains and almost anything that
moves. I'm not so keen on sprouts or the Welsh.</p>

  <p>My Grandma has a dog, called Poppy. She's small and cute and she
widdles when she gets excited.</p>
  <p>When I grow up I want to be like my Uncle... what a guy he is...</p>
 </div>
</div>
```

3. CSS

现在看看CSS。要做的就是删除原来的图像并用一个完全不同的图像替换它，我们是用单个图像来实现鼠标掠过的效果。

```
.box {
 position: relative;
}

html > body div#photo-lucas img {
 left: -5000px;
 position: absolute;
}

html > body div#photo-lucas {
 background: transparent url(lucas-rollover.png) no-repeat top left;
```

```
  height: 124px;
  left: 185px;
  position: absolute;
  width: 160px;
}

div.box:hover div#photo-lucas {
  background-position: top right;
}
```

4. 图像

图2-23是最初在XHTML中引用的JPG图像。

图2-24是一个PNG图像，用它替代原始图像。使用PNG图像是因为我们希望得到一个半透明的阴影，JPG和GIF文件都无法实现这一点。

图2-23　最初在XHTML中引用的图像(lucas.jpg，150px × 100px)

图2-24　用于翻转的图像(Lucas-rollover.png, 320px × 124px)，透明部分用格状图案表示

5. 所要做的工作

XHTML不值得仔细研究，但需要对CSS进行仔细分析。

图2-25显示的是起点。

首先必须删除原始图像(licas.jpg)。用CSS有很多方法可以做到这一点。但是最好的方法是设置其position属性为absolute，然后尽可能把它放在屏幕之外(如图2-26所示)。

```
Html > body div#photo-lucas img {
  left: -5000px;
  position: absolute;
}
```

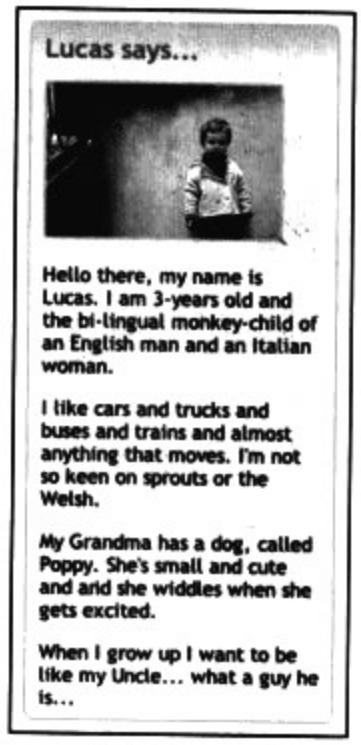


图2-25 起点

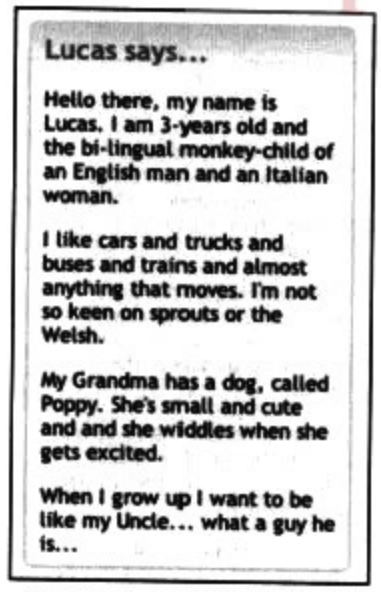


图2-26 删除原始图像

用position: absolute而不是position: relative的原因在于，它可以使一个按绝对方式定位对象从文档流中完全删除。在本例中，它使父元素(div#photo-lucas)消失，并使其下面的文本上移以填充现在是空白的区域。

清除了旧图像后就可以插入新的图像了。我们把它作为div#photo-lucas的背景图像，这个div包含原始图像，如图2-27所示。

```
html > body div#photo-lucas {
 background: transparent url(lucas-rollover.png) no-repeat top left;
}
```

插入的图像在哪里？

记住，它是div#photo-lucas的背景图像，而元素div#photo-lucas包含原始图像。但由于我们把原始图像扔到屏幕之外，因此div#photo-lucas没有内容，没有内容就没有尺寸。所以新的背景图像确实在那里，只不过必须给div#photo-lucas设置一个宽度和高度才能看到，如图2-28所示。

```
html > body div#photo-lucas {
  background: transparent url(lucas-rollover.png) no-repeat top left;
  height: 124px;
  width: 160px;
}
```

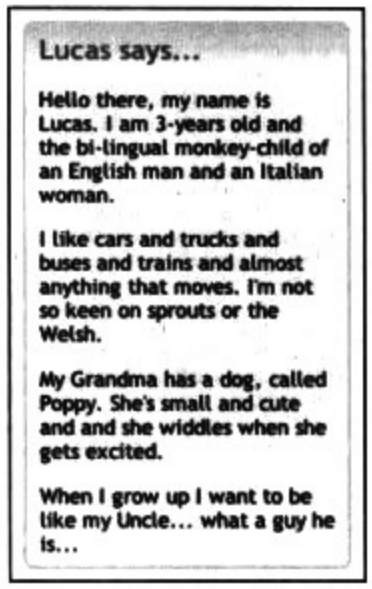


图2-27　插入新图像

图2-28　显示隐藏的图像

图像出来了，非常棒！注意，设置的高度要与新的翻转图像(lucas-rollover.png)的高度一致，而宽度正好是该图像宽度的一半。这样一次只能看到该图像的一半，这正是我们所希望的。

新图像将显示在屏幕上，且要把它重新定位在偏右的位置。要实现这一点，要再一次使用position: absolute(如图2-29所示)。

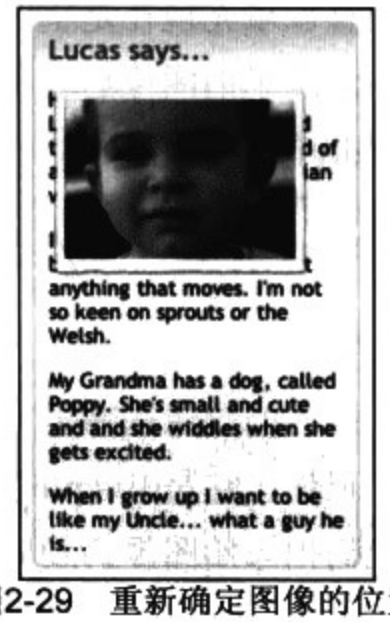


图2-29　重新确定图像的位置

```
    html > body div#photo-lucas {
     background: transparent url(lucas-rollover.png)
no-repeat top left;
     height: 124px;
     position: absolute;
     width: 160px;
    }
```

现在不必担心出现错误。所发生的是div#photo-lucas已从文档流中被删除，这意味着它不再与周围的对象相互影响。取而代之的是它悬浮在任何对象之上(或之下，如果用的z-index)并能定位在屏幕的任何位置。

因为在本例中要把它移到框的右边。首先要对原始框CSS稍做修改并添加下面一条规则：

```
.box {
  position: relative;
}
```

这意味着，无论给div#photo-lucas什么样的XY坐标，都是相对于包含它的元素(.box)进行定位的。左上是box的左上角，右下是box的右下角，等等。

既然建立了参考点，就可以移动图像了。首先设置左边的位置，如图2-30所示。

```
html > body div#photo-lucas {
 background: transparent url(lucas-rollover.png) no-repeat top left;
 height: 124px;
 left: 185px;
 position: absolute;
 width: 160px;
}
```

非常完美。我们不需要设置上边的位置，因为它已经处于正确的位置了。剩下要做的就是激活翻转器，这是该演示的重点，如图2-31所示。最后一步总是这么简单。

```
div.box:hover div#photo-lucas {
  background-position: top right;
}
```

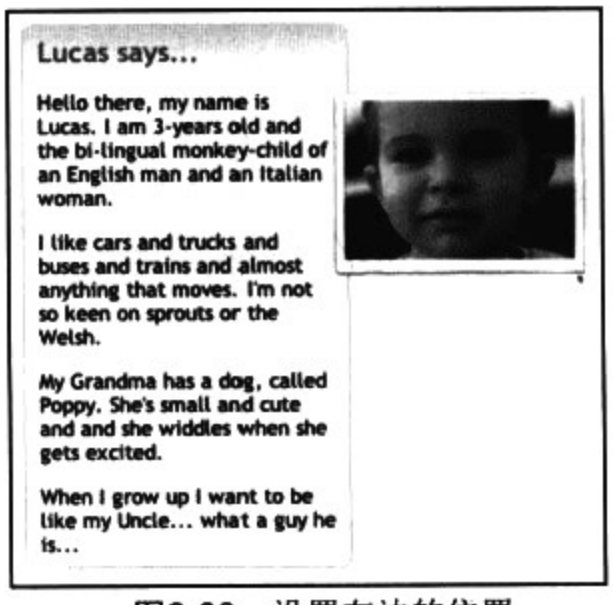


图2-30　设置左边的位置

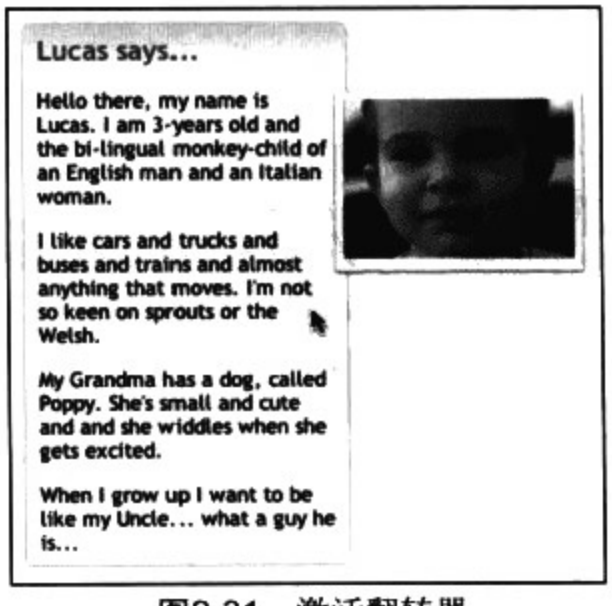


图2-31　激活翻转器

6. 对IE的处理

IE在这方面就显得相当惭愧。它不仅不解释div的:hover，而且也不处理半透明的PNG图像。因此用一个子选择符可以阻止IE实现这些改变，这在前面已提及。

确实有这样的实例，可以对IE 6/Win使用这种技术。我们可以用一个正常的JPG图像代替PNG图像(这样得不到阴影效果，但不一定是坏事)。只要初始图像位于链接内，翻转器也会工作(受一定限制)。

您可能会说："停止介绍在IE中不能工作的示例！"。然而我们应该从中吸取的是一种思想，即对当今最好的浏览器来说什么是可能的。一旦认识到什么是可能的，就可以开始调整该示例直到到达很好的平衡——CSS和跨浏览器功能之间的平衡，瘦的、普通的XHTML与在客户机器上使站点能工作这两者之间的平衡。

好了，我们先在IE 6中演示这个示例。需要修改的代码已用粗体字标出。所要做的就是为IE 6插入一个图像(一个JPG图像)，然后为Firefox、Opera和Safari替换这个图像。和往常一样，我们用子选择符来实现。

```
.box {
  position: relative;
}

div#photo img {
 left: -5000px;
 position: absolute;
}

body div#photo-lucas {
 background: transparent url(lucas-ie.png) no-repeat top left;
 border: 2px solid #e1d4c0;
 height: 113px;
 left: 175px;
 position: absolute;
 width: 156px;
}
```

```
html > body div#photo-lucas {
 background: transparent url(lucas-rollover.png) no-repeat top left;
 border: none;
 height: 124px;
 left: 185px;
 position: absolute;
 width: 160px;
}

div.box:hover div#photo-lucas {
 background-position: top right;
}
```

图2-32显示的是只在IE中使用的新图像。

所有一切在IE中是怎么样的呢？图2-33是完成后的IE产品。

图2-32　lucas-ie.jpg 文件，156px × 113px

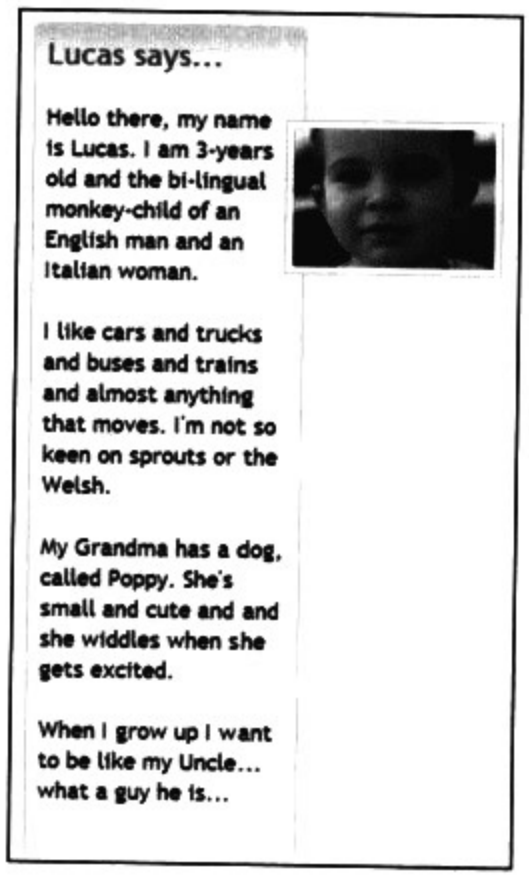


图2-33　完成后的IE产品

2.3 小结

本章研究了控制文本和背景颜色的翻转器，介绍了控制背景图像位置的翻转器。我们还注意到由于IE不完全支持:hover，从而使翻转器在IE中失效。

通过对本章的学习，我们了解了本章所介绍技术的基本原理，而不是仅留下一些盲目复制的代码示例。如果您不能把这些思想运用到客户的站点，至少您应了解这些思想并可以把这些技术复制到您的工作中。看到对一组代码进行如此简单的剪切和粘贴就可使之工作，您应感到很惊异。

下一章我们研究站点The PGA Championship的内部工作原理，这是一个成功的以事件驱动的网站。

第3章 经典的美国职业高尔夫联盟锦标赛网站

PGA锦标赛是世界上最重要的体育赛事之一。该赛事由美国职业高尔夫联盟举办，在每年的夏季末举行。该锦标赛是每赛季的最后一项重要赛事。吸引了世界最好的高尔夫职业运动员，并有世界各地成千上万的高尔夫爱好者追随。

Turner Sports Interactive是Time Warner的一个部门，负责该站点的开发和锦标赛期间的内容编辑。技术目标是用CSS(设计所有布局和表现)、易于理解的XHTML标记和Flash(用于实现某些特殊功能)创建一个动态的、内容丰富的、与标准兼容的网站。该网站的创作目标是追求独特的视觉效果、精美的表现，最重要的是不落入任何典型高尔夫网站设计的俗套。所选择的不饱和中性暖色调外加黑白色彩突出了原始图片，可使用户把更多注意力放在文本内容上。

PGA锦标赛网站于2004年7月发布，很快就受到高尔夫球迷和Web开发界的一致好评，其独特的表现方式、对Web标准的遵循和整体设计都得到了广泛认可。在比赛的第一天，由于PGA.com的编辑组不间断地发布新闻、比分和多媒体内容，导致网站流量激增。

自从这个版本的PGA锦标赛网站发布几年后，PGA陆续发布了几个不同的版本。但是我们认为只有2004年的版本是最好的。由于其小巧的CSS/XHTML标记组合、紧凑的设计、精美的高尔夫课程图片，使得它是非常经典的。

本章将详细介绍该网站采用的一些技术，正是这些技术实现了该网站最初的几个特色

功能。其中包括一些常用的CSS/XHTML技巧，也包括一些在自己工程中应注意的问题。本章介绍的主要内容如下：

- 用CSS和Photoshop创建分层的阴影效果
- 添加由CSS驱动的下拉菜单的最简单有效方法
- 在不违背标准兼容的情况下嵌入Flash内容

3.1 阴影效果的实现

PGA锦标赛网站最吸引眼球的特色之一是其几近三维(3D)的内容分层。效果极细微，但如果仔细观察如图3-1所示的主页(http://www.pga.com/pgachampionship/2004/index.html)，会发现左边一栏是悬浮在其周围内容之上的。这不仅是很酷的视觉技巧，也起到了一定的编辑目的。左边一栏包含最新的锦标赛新闻和特色。给这一区域一个细微(但能察觉)的视觉提升感觉，最终用户很快就能发现那里发布的是最新的锦标赛内容。

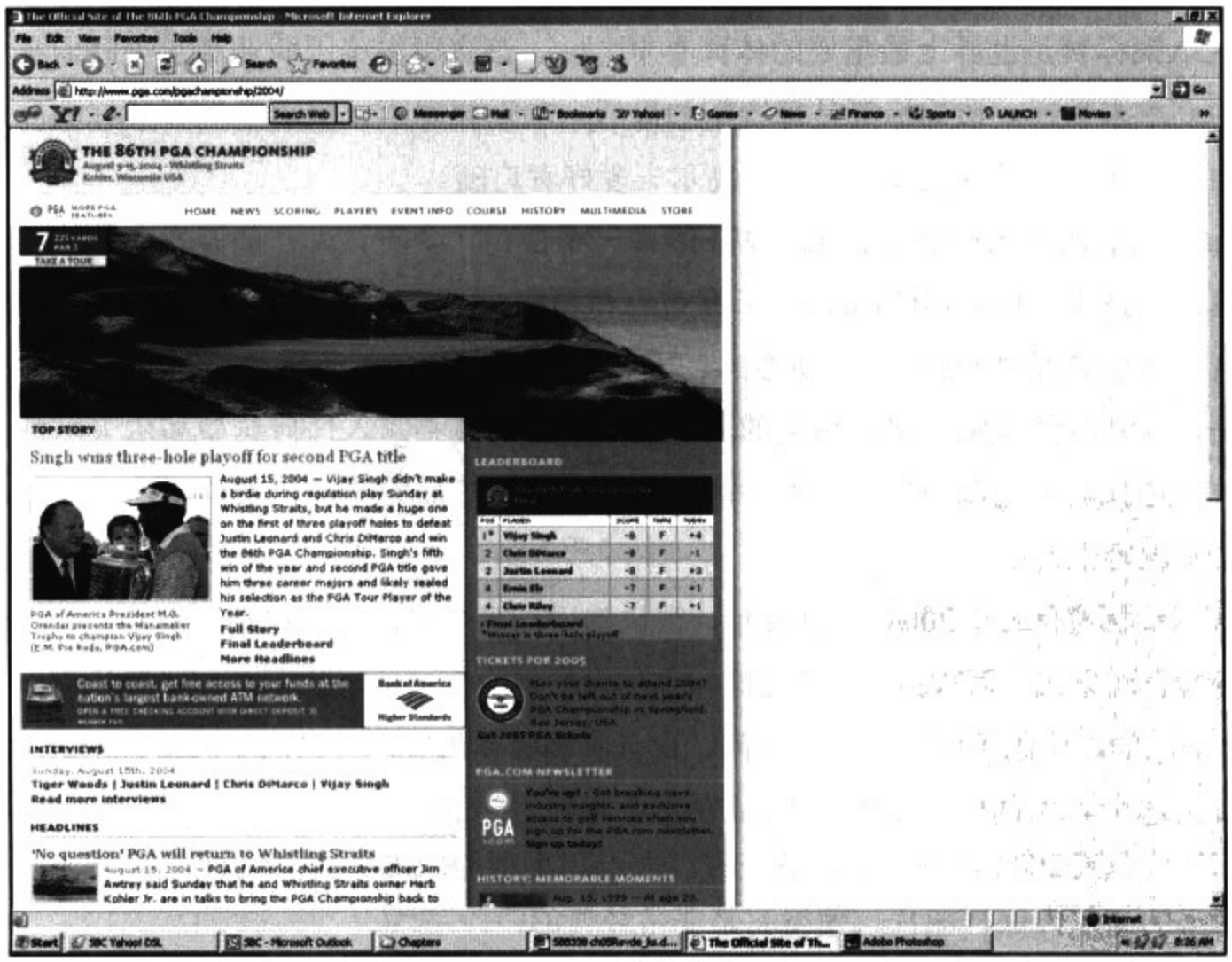


图3-1 PGA锦标赛主页

左边一栏不是简单的比周围内容“高”一些，而是需要使这些内容有一定的模糊，使得这一区域似乎真的悬浮在页面上。奥秘就在左边一栏上面的Top Story题头上。这个题头似乎与左边一栏相连，并在右边有一逐渐从页面淡去阴影，这样就感觉它是这个XHTML的一部分，位于Flash动画的上面。但这只不过是个幻觉。

整个效果看起来似乎很复杂且要求很高的带宽，但实际上很小(大小仅以字节计)，用一个提前编制的计划就可相当简单地实现。本节介绍用Photoshop、Flash、CSS和XHTML实现这一效果，并介绍能更好地实现该效果的另一技巧。

3.1.1　创建幻觉效果

在介绍具体的实现细节之前，再回顾一下我们的目标——使页面上部的Flash动画和下面的内容产生视觉上的混合效果，还要有左边一栏比周围内容高一个层次的效果，后一效果是通过一个从文档逐渐淡去的阴影实现的。这需要技巧，因为在Flash动画或XHTML中1px的偏差都会破坏这种效果。但由于CSS和XHTML的精确性，这是可能实现的。

和任何嵌入内容一样，Flash动画总是方形的。要嵌入一个不规则形状的Flash对象是不可能的。要“打破边界框”和创建分层的视觉效果，左边一栏的部分设计应位于Flash动画的底部之内。当Flash动画和XHTML内容紧挨在一起时，这种布局(按像素)就使它们可共同形成这种幻觉效果。

首先要在Photoshop中创建一个文档，并以一个十六进制的颜色进行填充，Flash动画和从页面右边逐渐淡去的左边栏采用的也是这种颜色。这样就创建了一个新层(命名为bar)。然后，选择与左边一栏宽度一致的空间(用工具Rectangular Marquee选择)，并以选择的颜色填充这部分区域。把一个阴影层效果应用于该bar上，设置一些阴影选项使产生的外观不是很明显，但足以产生一种分层的细微效果。

完成上述操作后该文档看起来如图3-2所示。

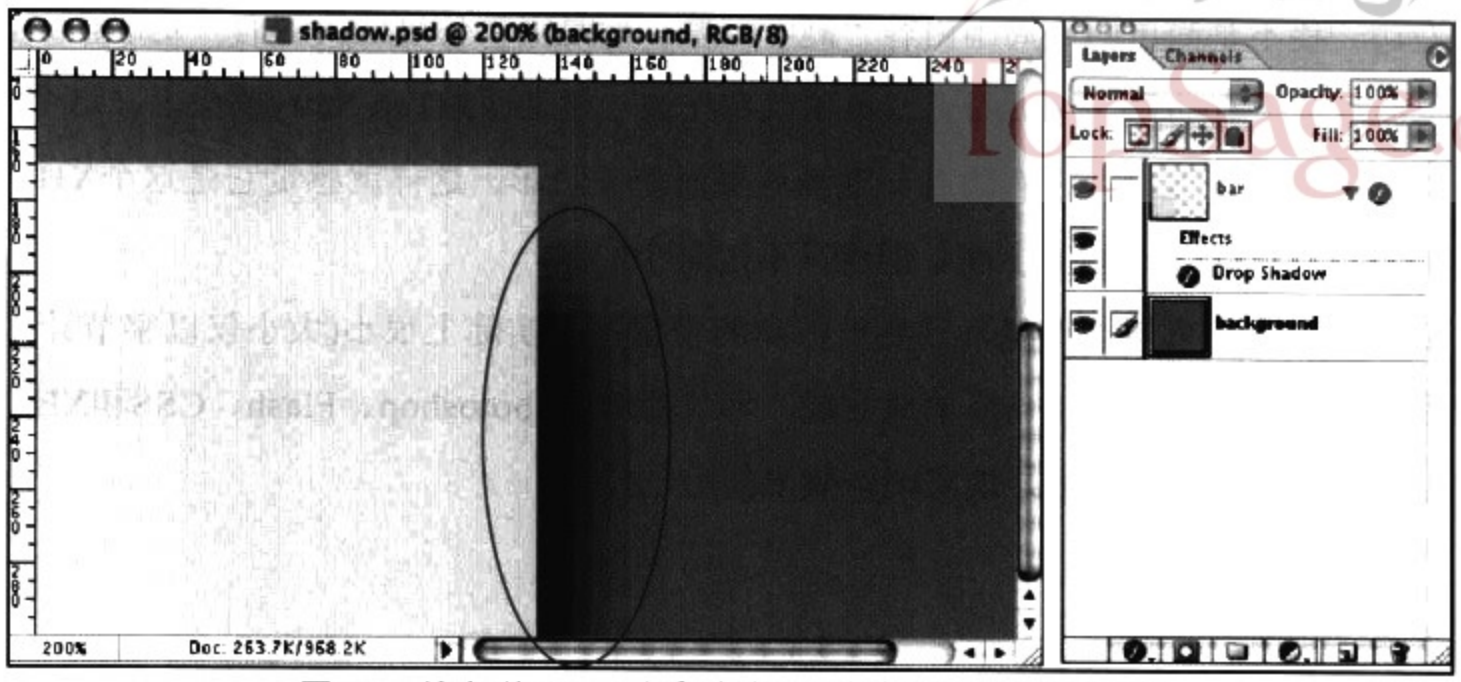


图3-2　放大到200%(以看到阴影)后的初始Photoshop文档

一旦阴影在右边出现，就应隐藏背景图层，并用工具Rectangular Marquee选择阴影本身(如图3-3所示)。把阴影复制到剪贴板(Edit→Copy Merged)，这就创建了一个含透明背景的Photoshop文档，阴影被粘贴在文档里。把图形存为PNG文件并放在Flash动画(这里已有一个矢量图，其高度和宽度与Photoshop文档中的bar的高度与宽度相同)里。

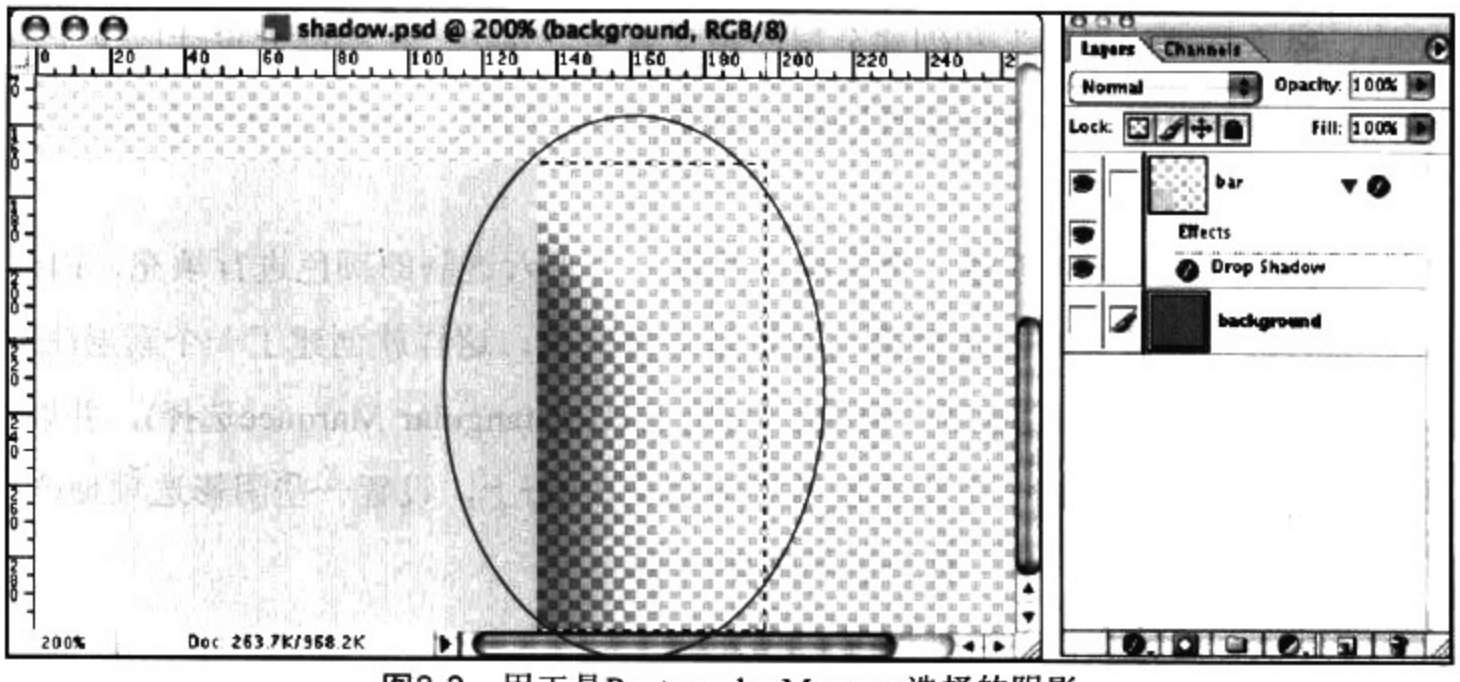


图3-3　用工具Rectangular Marquee选择的阴影

您可能会问，为什么把图形保存为PNG文件而不是更常见的GIF或JPEG文件？这是因为，把位图导入Flash时，PNG是首选格式。为什么？因为PNG文件格式提供了含256个透明层次的alpha通道。这样就能从您偏爱的位图编辑器(Photoshop、Firework等)中导出更细

致的图像、把它们导入Flash、发布动画。这些动画在同时采用有损JPEG压缩时能保持图像的透明数据。这样既具有PNG文件格式的视觉质量，文件大小又与一个JPEG文件差不多，因此是最好的。

Flash动画完成后，接下来就该用CSS/XHTML复制阴影了。由于阴影要一直持续到文件的长度，因此只要有内容，浏览器就不得不自动重复这个阴影效果，这就需要图形具有可复制的部分。这要求图形能垂直叠放(一个挨一个的重复复制)且要无缝、无间隙和不失真。因此选择1px高的区域，确保包含一些用于平滑融合的背景色，如图3-4所示。

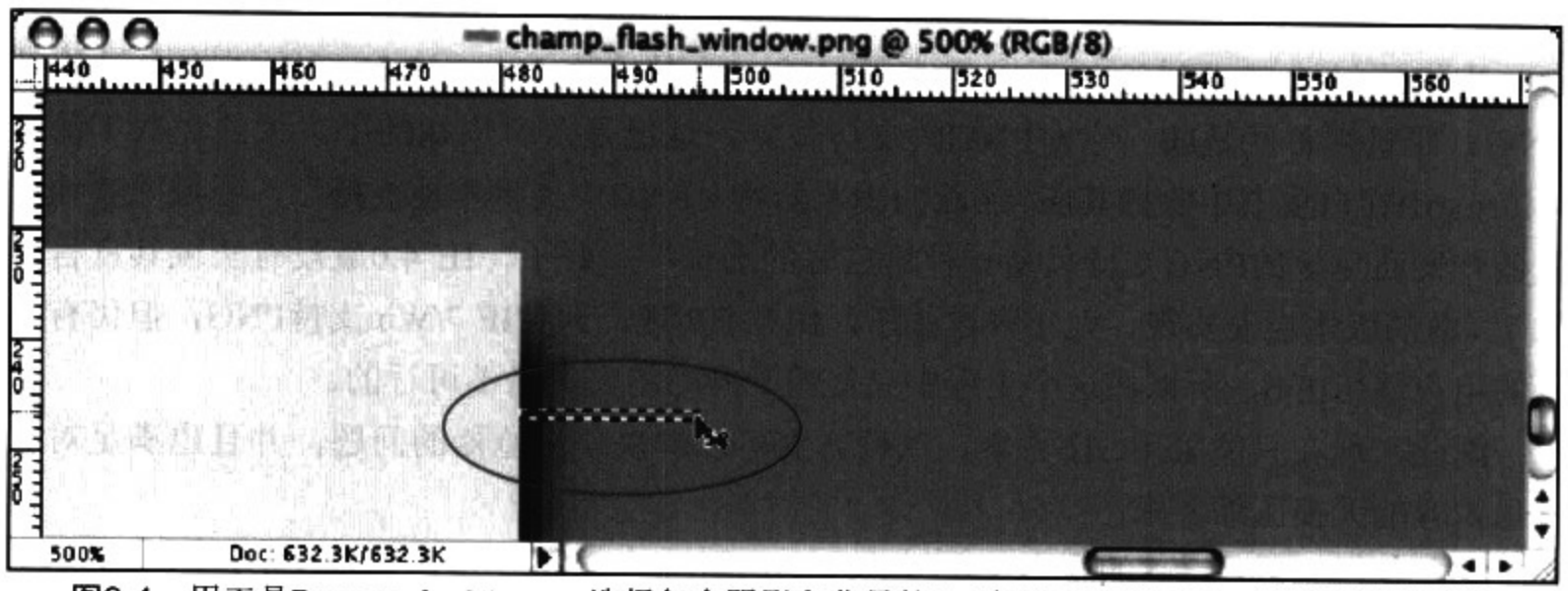


图3-4　用工具Rectangular Marquee选择包含阴影和背景的1px高的可重复区域，并复制到剪贴板

一旦复制了该区域，就创建了一个新文档(Photoshop自动设置剪贴板的宽度和高度)且已把所选区域粘贴在内，如图3-5所示。

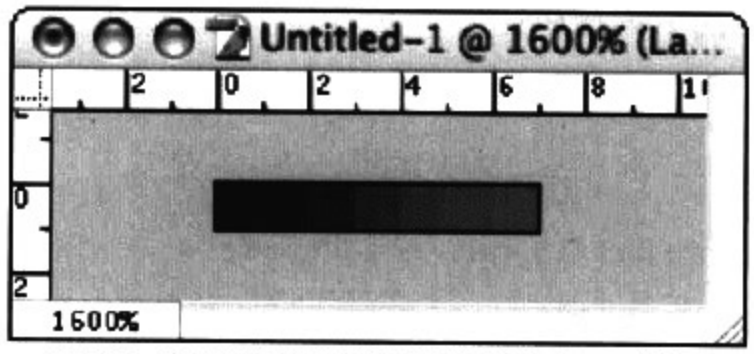


图3-5　粘贴到新文档的剪贴板内容(已放大)

这里关注的是阴影本身，但也必须关注左边栏的浅灰色背景。为此，把图形的画板大小增加到所要求的像素宽度，阴影右对齐。如图3-6所示，透明区域已用左边栏的十六进制颜色填充，把整个图形保存为一个GIF文件。

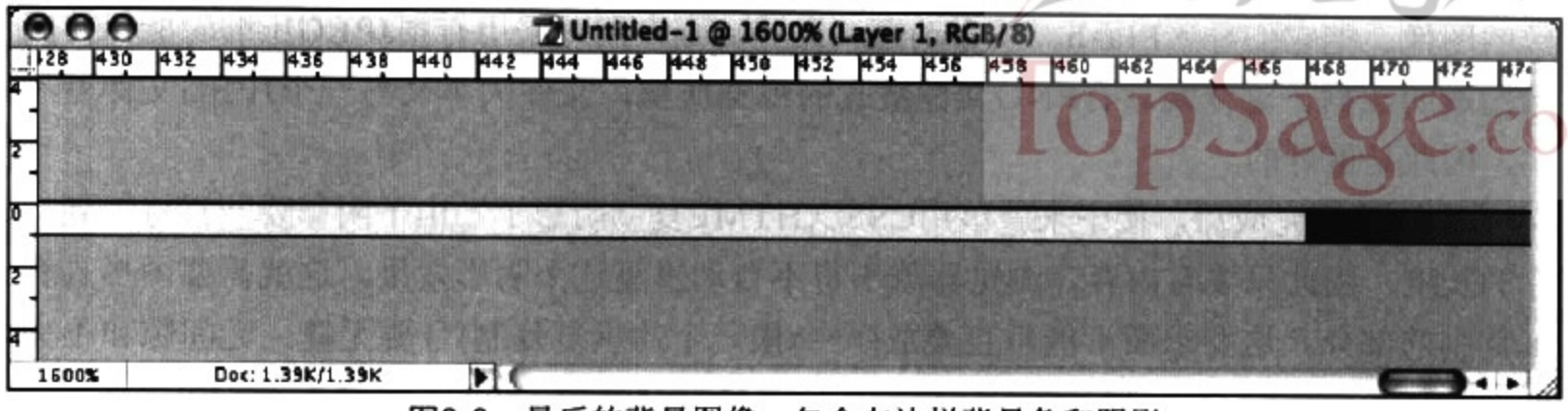


图3-6　最后的背景图像，包含左边栏背景色和阴影

我们回过头来再讨论一下图像文件格式。为什么在这里用GIF而不用前面讨论的PNG？罪魁祸首就是IE，它对PNG的支持很差，这已是众所周知的了。大约在八年前，Microsoft在白皮书中吹捧其IE 4.0版的浏览器将对PNG“提供本地支持”，不像“其他浏览器开发商包含的PNG支持只是一种第三方的选择”。好了，IE 4.0版没有实现其宣言，IE 7以前的版本也没实现，九年多过去了，IE 7才实现。即使IE 7/Win支持PNG，但仍有大量的用户使用IE 6，所以在这个工程中只依赖于PNG的方案是不可行的。

因此，现在只能采用GIF方案，这种方案可以解决跨浏览器的问题，并且也满足对非图形内容的无损压缩要求。

现在我们来看看CSS的魔力。在样式表中，用div创建了一个“容器”，这是一个不可见的块，它为内容组(在本例中是左、右两栏)的定位提供了一个外层框架。随着任一栏内的内容向下增加，这个容器就相应进行扩展。这就是为什么把背景图像应用到这个容器而不是栏的原因——这样就可以不管里面内容的长度而对图像进行重复。要在CSS中实现这一点，需要在样式表中添加下列代码：

```
    #colwrap {
    width:740px;
    background:#96968C
    url(http://i.pga.com/pga/images/pgachampionship/img/bg_home_content.gif)
repeat-y;
    }
```

给div设置一个指定的像素宽度，后面是一个背景属性(这是所有行为发生的地方)。我们用一个与Flash动画和右边栏背景一致的十六进制值。右边一栏是图像的url和指令repeat-y，该指令指示浏览器为整个div长度按纵向重复图像。

没有repeat-y指令，浏览器(默认)不仅在垂直方向也在水平方向上层叠图像。这就导致图像的左边又一次出现在阴影的右边。这显然与设计不相符，因此用repeat-y指令使浏览

器只在一个方向(向下)重复图像。如果设计要求水平重复图像，可用repeat-x指令。

由于CSS允许把一个十六进制的颜色和一个图像应用到单个元素，因此可用CSS对颜色一致(就是所谓的打印术语“专”色)的区域着色，这可以削减文件的大小，实现起来也非常简单。这就是在阴影边上剪切掉容器背景图像的原因。背景其余部分(阴影融合的深灰色)用浏览器规定的颜色着色。虽然这种方法并不是必需的(所有一切可以包含在背景GIF文件中)，但这样可以节省一些字节。我们都知道，每节省一位都是有好处的。

完成这些工作后的div如图3-7所示。

div背景完成后就把Flash动画添加到容器div之上，这两个部分结合得非常完美，如图3-8所示。

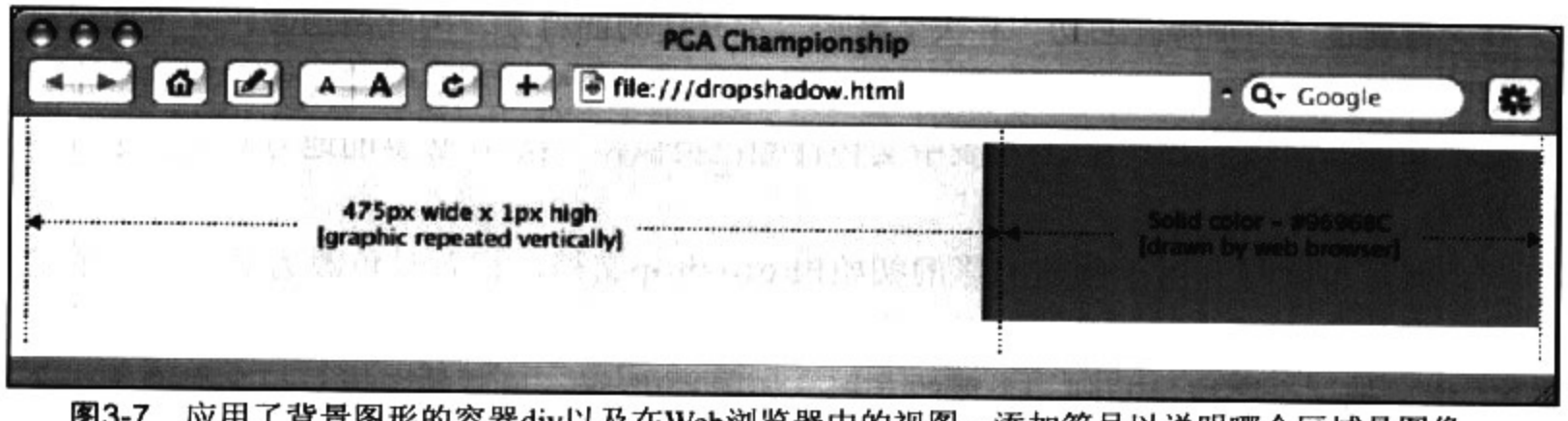


图3-7　应用了背景图形的容器div以及在Web浏览器中的视图：添加符号以说明哪个区域是图像，哪个区域是由浏览器绘制的

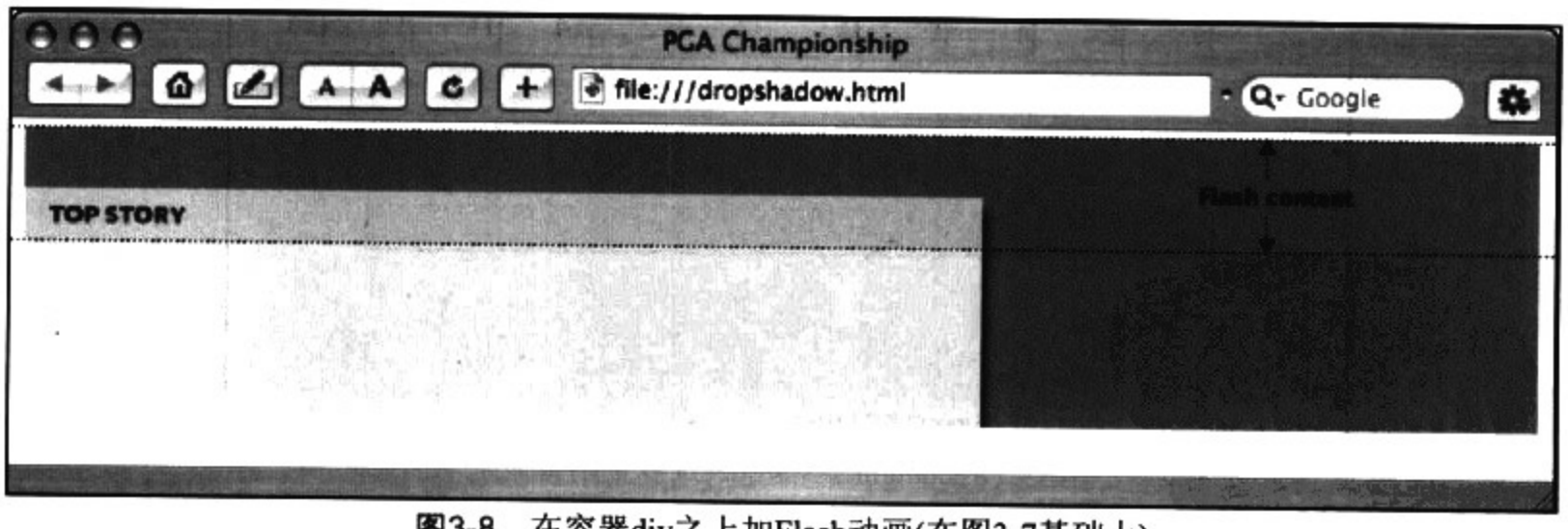


图3-8　在容器div之上加Flash动画(在图3-7基础上)

在向下继续之前，介绍一个小技巧。为了保证设计是超前的，采用网格来显示元素的像素快照是更合适的。这种效果要求事先知道内容的准确位置。如果您要超越自我而试图应用这样的效果，仍然要考虑布局，这可能不得不多次返工。为了实现这样的效果，就应该把设计工作过程中的这种反复想像为“甜蜜的”阶段。一定的耐心和计划可以减少很多

痛苦并节省大量的时间。

3.1.2 使阴影更真实

现在介绍一个基本技巧，利用这个技巧可以给前面创建的阴影效果添加额外的视觉定义，使阴影看起来更真实。

在现实世界(是的，显视器之外的地方)里，阴影的不透明度根据它所遇到的表面不同而变化。如果表面是白色的，阴影就表现为不同色调的浅灰色。如果表面很暗，阴影就更暗。这是非常明显的事实，但在计算机屏幕的平面世界里，就不是那么直接了。

在PGA锦标赛网站，根据阴影与不同背景色的区域接触情况来控制其外观，从而使这一概念得到进一步加强。右边一栏大多数内容都有透明的背景，因此所用颜色是下面已有的颜色，但是新闻链接就有其特殊性。由于没有视觉分离感，用户很难分辨出新闻链接。因此把其他新闻项的背景变暗(与表中交替行颜色类似)，然后在必要的地方对阴影进行一些处理。

为此，再次打开用于创建阴影的初始Photoshop文档，把背景色改为更暗一些的值(#828279)，使整个布局更协调。选择阴影中间1px高的区域(与前面主区域阴影的做法一样)，然后把这个区域复制到一个新的文档。如图3-9所示，最后的图形看起来很像第一个图形，只不过更暗一些。

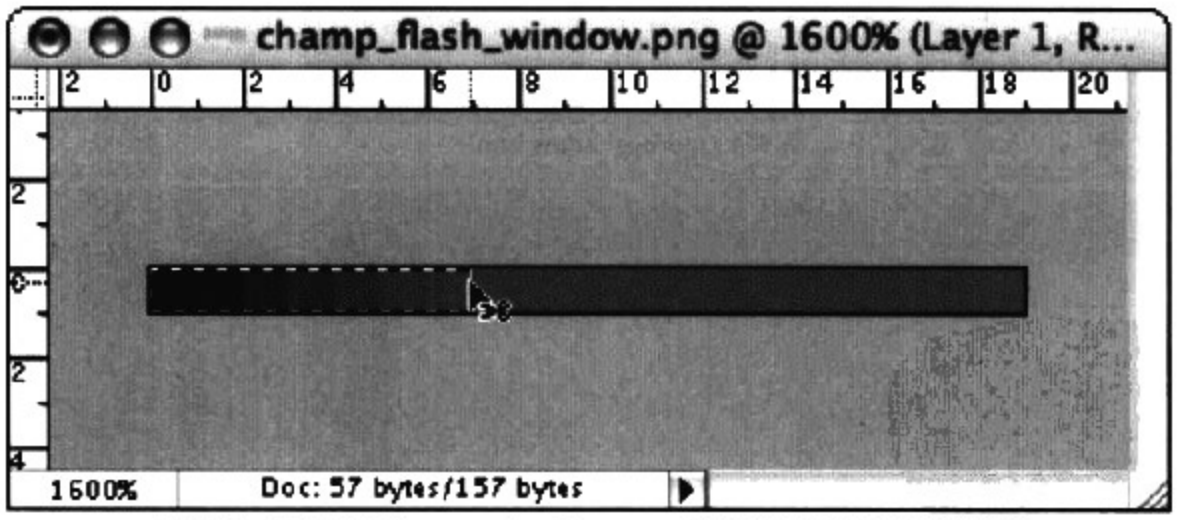


图3-9 用工具Rectangular Marquee选择更暗背景上的阴影

对于XHTML，要创建一个无序列表来标记内容。在语义上，列表元素非常适合包装杂乱的数据和导航条(在本章稍候将看到)。在用一个样式表进行控制后，就可以提供各种各样的视觉表现。

因此，可在XHTML模板的右边一栏创建一个新闻项的无序列表，代码如下所示：

```
<ul class="stories">
 <li>DiMarco and Riley play their way into Ryder Cup</li>
 <li>'No question' PGA will return to Whistling Straits</li>
 <li>Sullivan lowest club professional at PGA since 1969</li>
 <li>PGA of America adjusts Sunday yardages at Whistling Straits</li>
</ul>
```

给这个无序列表元素分配的类是stories，然后把这个类添加到样式表：

```
ul.stories {
 margin:0;
 padding:0;
 color:#E9E9DF;
}
```

首先，stories被指定为无序列表元素的子类。然后重置Web浏览器应用于无序列表元素的默认属性。这是通过把内、外边距设置为0来实现的。然后添加一个颜色属性，这个属性影响列表内的所有文本。

说明：

从技术角度讲，可以去掉ul而创建独立的类stories。然而，把类直接分配给HTML元素不仅是一种好的形式，而且也使样式表更易读。把元素看作是内联的注释，一眼就能看出它们所描述的功能，无论是几个月后再来编辑该样式表还是多名开发人员共享该样式表，都能很容易看出该类所属的元素。前面所付出的少量组织工作将节省大量的时间。

处理完无序列表对象后就该处理其中的每一个列表元素了：

```
ul.stories li {
 list-style:none;
 margin-bottom:2px;
 padding:4px 4px 4px 10px;
}
```

我们将一行一行地分析。首先list-style被设置为none，这将阻止浏览器在列表项前添加圆点的默认行为。然后添加少量外边距以增加列表项之间的纵向距离，再设置内边距(上、右、下边距各为4px，左边距为10px)。

默认情况下，无序列表stories内的每个列表项都将采用这些值。在这个阶段，它们都有相同的背景(用其下元素的颜色)，但是在这里这个额外的效果产生了作用：

```
    ul.stories li.odd {
     background:#828279
    url(http://i.pga.com/pga/images/pgachampionship/img/bg_stories_shadow.
gif) repeat-y;
    }
```

通过继承，类odd预先加载前面所分配的所有属性，只留下需要修改的属性——background。现在分配的背景色更暗，是一个用十六进制表示的值。然后提供了背景图形的url和一个重复指令，这一指令指示浏览器纵向重复背景而不是水平重复。

把无序列表代码加入XHTML，把odd类应用于每个奇数列表项(这是手工完成的，虽然也可以用JavaScript、PHP等编程实现)：

```
<ul class="stories">
 <li class="odd">DiMarco and Riley play their way into Ryder Cup</li>
 <li>'No question' PGA will return to Whistling Straits</li>
 <li class="odd">Sullivan lowest club professional at PGA since 1969</li>
 <li>PGA of America adjusts Sunday yardages at Whistling Straits</li>
</ul>
```

所有这些完成后，无序列表看起来是右边一栏的一部分且位于主内容区的阴影之下，但实际上它位于前面创建的背景之上(如图3-10所示)。技巧就是直接把这个右边栏(包含无序列表)部署在左边栏的右边缘之上。这就造成一种幻觉，以为列表项更暗一些的背景色是页面上已存在阴影的一部分，但实际上它在阴影的上一层。

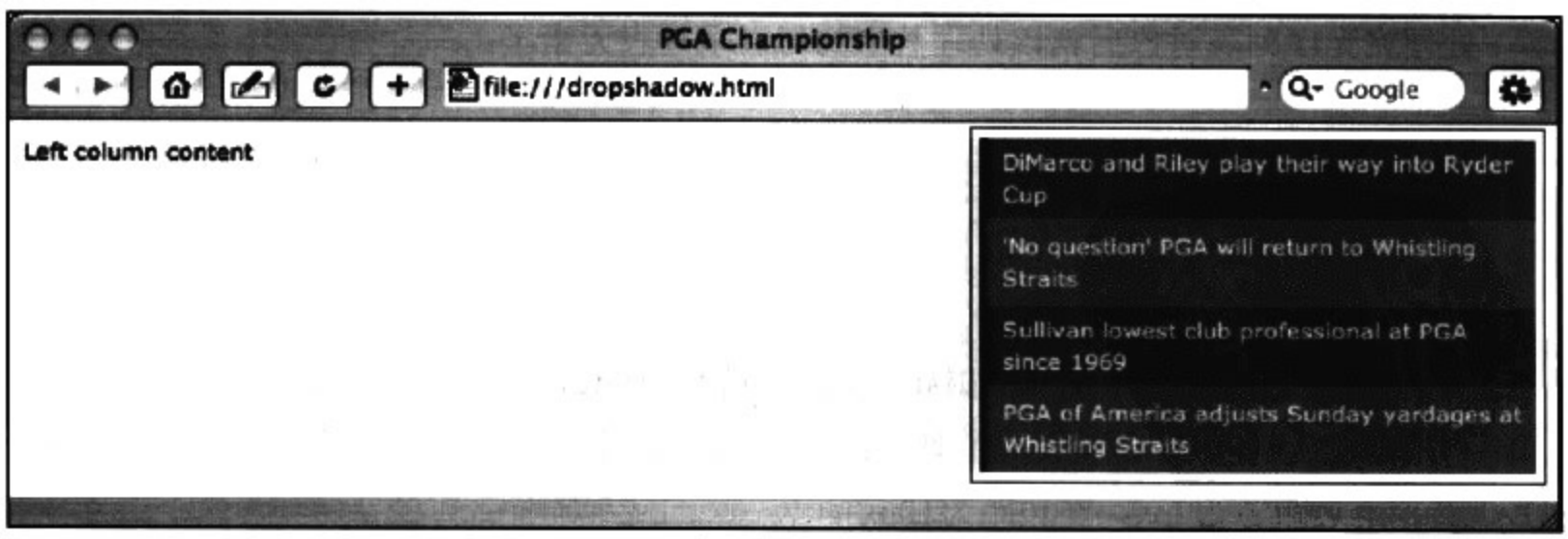


图3-10　XHTML div容器中具有更暗阴影背景的列表项

下面是关于左、右两栏的CSS：

```
#lcol {
```

```
 width:468px;
 float:left;
}
#rcol {
 width:271px;
 float:right;
}
```

创建这种效果所需的主要XHTML如下：

```
<div id="colwrap">
 <div id="lcol">
  <! - Left column content - >
 </div>
 <div id="rcol">
  <ul class="stories">
   <li class="odd">DiMarco and Riley play their way into Ryder Cup</li>
   <li>'No question' PGA will return to Whistling Straits</li>
   <li class="odd">Sullivan lowest club professional at PGA since 1969</li>
   <li>PGA of America adjusts Sunday yardages at Whistling Straits</li>
  </ul>
 </div>
</div>
```

这样，额外的阴影效果就完成了。

充分利用浏览器自动重复背景图像和把颜色应用到同一元素的能力，就有无数机会仅用一个效果就把视觉景深和丰富色彩加入到平淡静态的布局中。所有需要的不过是一定的耐心、少许规划和一些实验而已。

3.2　创建CSS下拉菜单

在20世纪90年代后期的.com淘金浪潮中，高新、前卫站点的标志就是经常融入一些奇特的导航菜单。但是炫目的外观后面常常是复杂的JavaScript嵌套、臃肿的HTML、甚至更糟的还有私有浏览器API方法。尽管站点的意图是使用户的体验更流畅、更直观，但早期的下拉式解决方案给我们带来的更多是沮丧(特别是操作失效时)和臃肿的文件。

然后出现了CSS和具有神奇魔力的:hover伪类选择符。像Eric Meyer和Christopher Schmitt这样的大师发布了一些指南，教大家如何使用一个锚标签的:hover属性。当:hover

属性与标准的ol无序列表一起使用时，就可创建一个下拉菜单，这种菜单与用常规方法创建的菜单在外表上相似，不过还是显得有点臃肿和复杂。

但尽管是纯粹的CSS菜单，仍存在一个巨大的问题：IE/Win。到目前为止IE/Win仍是访问Web最流行的浏览器，却不支持:hover属性(通常也不支持CSS，但那是另一回事)，因此它不能呈现这种下拉菜单。结果CSS菜单充其量被归为狂热爱好者的工具。

这种情况在2003年10月开始改变，Patrick Griffiths和Dan Webb用Suckerfish Dropdowns (http://www.alistapart.com/articles/dropdowns)激活了CSS界。Suckerfish Dropdowns是一个简洁的、CSS驱动的下拉系统，它几乎在所有浏览器上都能使用，包括IE/Win。Suckerfish不仅很小，而且跨浏览器兼容、遵从Web标准、语义丰富、可访问性好。

Suckerfish下拉菜单还非常容易创建。如果您知道如何在XHTML中创建一个无序列表，那么就掌握了一半。所有表现层和功能都由一个很小的样式属性集控制。

Suckerfish第一次发布几个月之后，Griffiths和Webb就把赌注下在Son of Suckerfish Dropdowns。这是比第一个版本更小的版本，具有更好的兼容性，可以导入多重下拉菜单。它有点像是Suckerfish的孩子，PGA锦标赛网站用的就是这种菜单，但是我们不对其基本结构做深入研究(可以从http://www.htmldog.com/articles/suckerfish免费下载示例)。本节将讨论如何定制、可能的陷阱和一些常用的使用技巧。

3.2.1 定制下拉菜单位置

在为PGA锦标赛网站的导航条修改Suckerfish时遇到的第一个问题是确定下拉菜单的位置。嵌套的下拉菜单默认是直接出现在父列表项之下，这取决于所含图形或文本的高度。在PGA锦标赛网站这个示例中，父图形被裁减得比周围区域短(以降低文件大小)。因此弹出的下拉菜单位于图形链接之下而不是在白色导航条(这是该网站最后的形式)之下。

可以在图形链接之下添加一些额外空白区域，使得在希望的位置弹出下拉菜单，但是这可能增加每个图形的文件大小并在没有内容的区域创建“热”链接。这个挑战的应对办法是找到一种方法，使弹出的下拉菜单位于空白导航条下面，但不会反过来影响或改变已有内容。

第一步很简单。每一个嵌套的无序列表(下拉菜单)已按绝对方式定位，因此添加一个

top属性，在需要的时候就可把菜单往下移：

```
#nav li ul {
 position:absolute;
 left:-999em;
 top:20px;
}
```

这就成功地把每个下拉菜单相对于其父列表项下移20px，下拉菜单也就位于空白导航条之下。但这又出现一个新问题。现在当鼠标指针向下移的时候，主链接和下拉菜单之间的区域脱离了翻转器，即这个区域成了不可单击的空白区域。因此下一步就是找到一种方法，只要鼠标指针进入这一空白区域就使下拉菜单可见。

在默认情况下，一个列表元素仅能与它所包含的内容一样高。但是我们可以用CSS进行修改：

```
#nav li {
 position:relative;
 float:left;
 margin:0 15px 0 0;
 padding:0;
 width:auto;
 height:20px;
}
```

这里重要的部分是height属性。通过指定一个高度(因此重写了默认的高度)，每个列表元素的不可见边界框就向下扩展，填满了空隙。列表项现在看起来就像包含了一个20像素高的图形，但实际上没有这么高。但浏览器能分出这种差别，因此下拉菜单就可如期弹出。

从图3-11可以看到列表项和图形元素的受影响情况。用Web Developer扩展可以添加一黑色描边，使列表元素和图形的不可见的边界框可见。Web Developer扩展是Chris Pederick为Firefox开发的，是免费的(http://chrispederick.com/work/web-developer)。这对样式表修改有一个直观的确认，并展现了浏览器的实际显示结果。在开发PGA锦标赛站点时这个扩展很多次都派上了用场，我向涉及Web开发和设计的每一个人推荐它。

图3-11　用Firefox看到的PGA锦标赛站点，用Web Developer扩展描出了所有列表元素和图像的轮廓

3.2.2　定制下拉菜单的样式

下拉菜单工作正常并在适当的位置弹出后，就该调整菜单项的外观了。

首先把嵌套无序列表的背景色设置为白色，并给所有的下拉菜单分配一个统一的宽度(基于最长的菜单标题)：

```
#nav li ul {
 margin:0;
 padding:0;
 position:absolute;
 left:-999em;
 top:20px;
 background:#fff;
 width:146px;
}
```

从图3-12很容易发现问题。每个下拉菜单的左边缘与它的父列表项的左边缘对齐，每个菜单项之间没有足够的视觉分离。

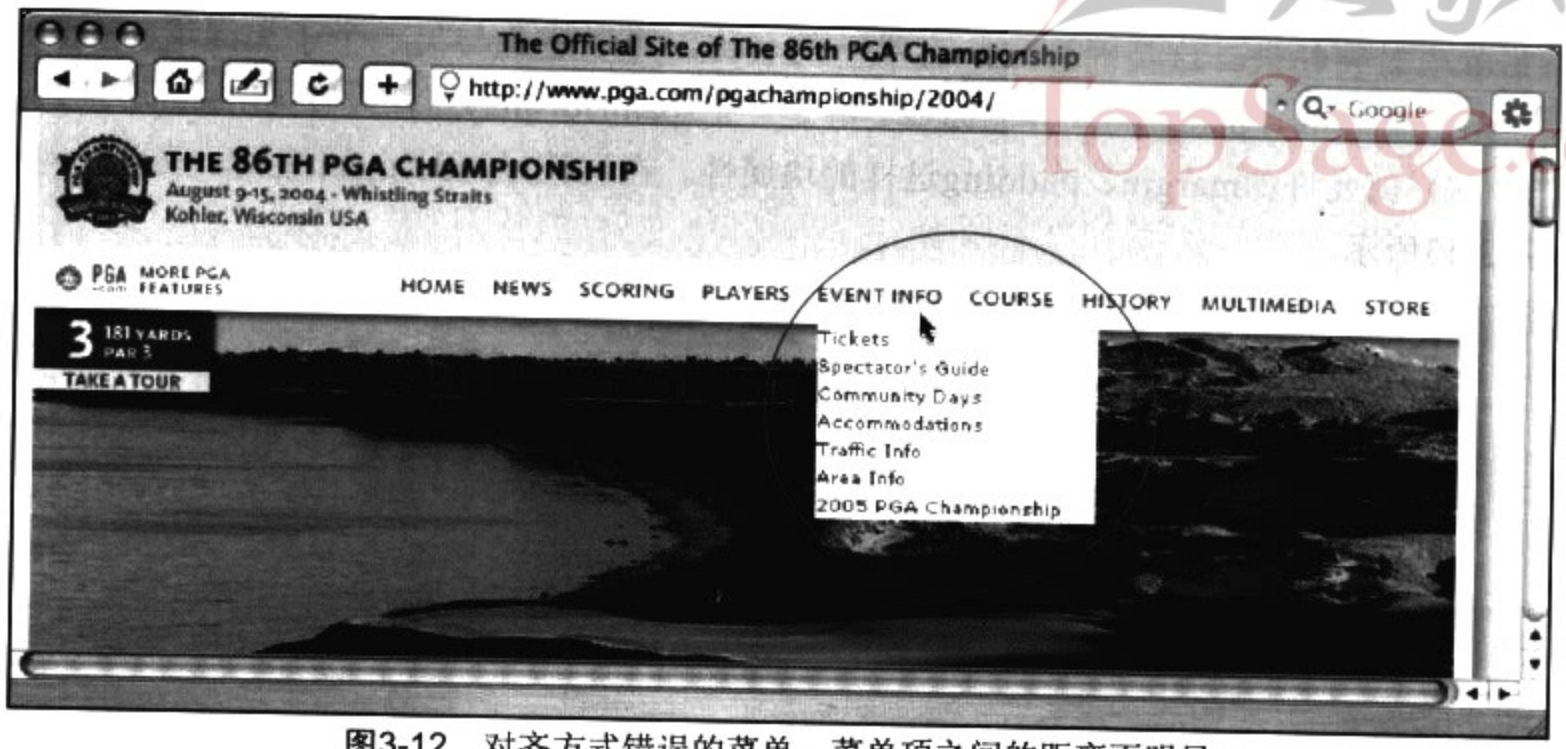


图3-12　对齐方式错误的菜单，菜单项之间的距离不明显

因此需要对嵌套列表项作额外的修改：

```
#nav li li {
 height:auto;
 margin:0;
 padding:0;
 width:100%;
 font-size:9px;
 border-bottom:1px solid #F5F5F0;
}
```

由于每个列表项的width属性被设置为100%，因此这些框架就向外扩展，达到父元素一样的宽度——在这里是146px。如果不修改列表项的默认宽度，浏览器所画的底边框就只与所含文本等长。设置列表的宽度为100%，菜单项就有了统一的外观，而不用考虑每个列表项包含多少文本。

接下来就该处理文本内容了：

```
#nav li li span {
 display:block;
 margin:0;
 padding:3px 4px 3px 7px;
 position:relative;
}
```

为了更好地控制菜单项中各个列表项内的每个文本块的位置，我们用span标签把每

个菜单项封装起来。这是因为span有唯一的名称，可以避免与另一个子类混淆。在语义上，span也比div、paragrph或其他标签更清楚。把span的display属性设置为block(不是默认值)，这样就允许如margin、padding这样的块属性。在调整好padding属性后，菜单就应该如图3-13所示。

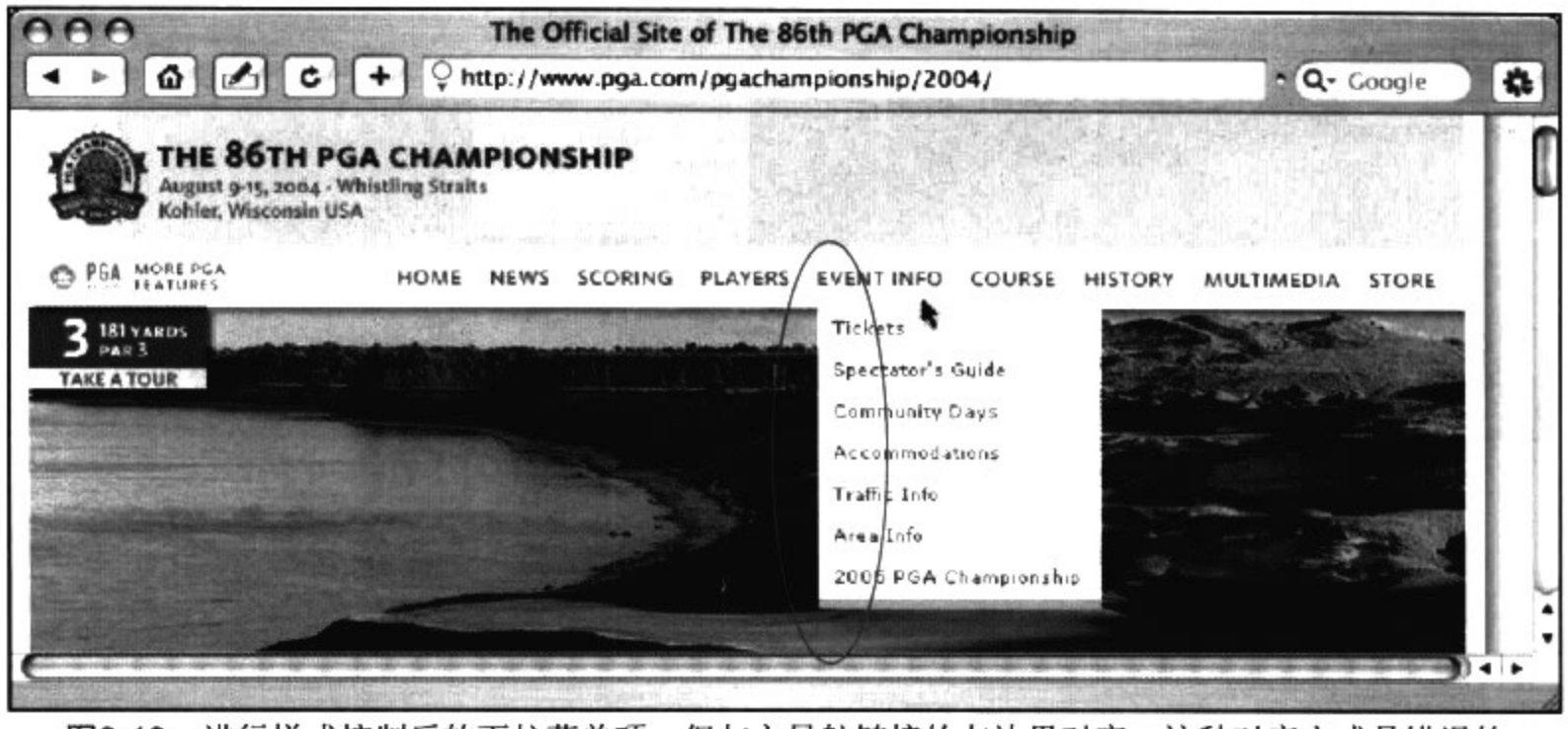


图3-13　进行样式控制后的下拉菜单项，但与主导航链接的左边界对齐，这种对齐方式是错误的

下拉菜单看起来已经完成了，但仍有一个问题。文本菜单项不再与它们对应的父列表项的左边缘对齐。虽然不是必须要修改，但向左边做略微调整看起来会更好。幸运的是，这与把每个无序列表强制在左边一样容易，如下所示：

```
#nav li ul {
 margin:0 0 0 -8px;
 padding:0;
 position:absolute;
 left:-999em;
 top:.0px;
 background:#fff;
 width:146px;
}
```

感谢这些负值！把左外边距从0改为-8(外边距值的顺序是上、右、下、左)，就把每个嵌套的无序列表左移8px(如果是正的则右移)。这就使每个文本菜单项的左边缘与父列表项完全在一条线上，如图3-14所示。

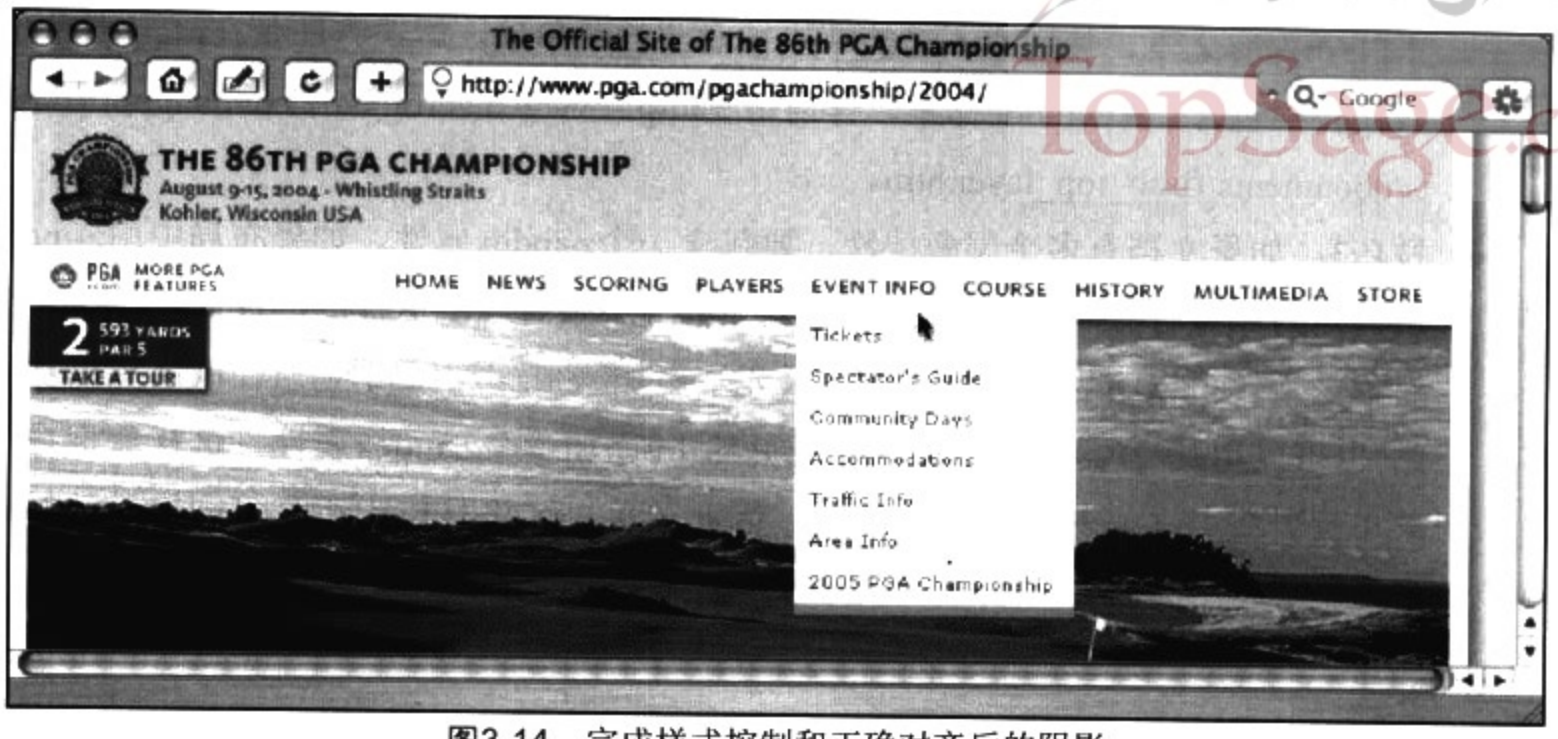


图3-14　完成样式控制和正确对齐后的阴影

1. 重要提示

既然我们已介绍了修改最初来源的方法，在使用Suckerfish下拉菜单时有一个潜在的问题要注意。在IE/Win中，Suckerfish需要JavaScript。这也是IE对CSS支持不够的一个实例。在Mozilla、Firefox和Safari中，Suckerfish可完全由CSS激活，但为了在IE/Win中能奏效，必须在使用这个下拉菜单的任一文档中添加一小段JavaScript。这个hack相当小，通过给文档对象模型(Document Object Model，DOM)附属一些定制行为，就可轻松解决IE不支持:hover伪链接元素的问题。但这有一个障碍，如果IE/Win用户关闭了JavaScript，则下拉菜单就不会出现。实际很少有人关闭JavaScript，但这是我们必须注意的一个问题。

2. 了解用法

解决了浏览器的支持问题，下面提供一些在自己实现CSS下拉菜单的工作中会用到的一些实用技巧。

- 技巧1：提供备份。由于前面提到的跨浏览器问题，在网站布局中包含一个次级导航菜单就显得很重要，在下拉菜单失效后次级导航菜单就能发挥作用。否则访问者就不能在网站内进行导航。对任何类型的下拉导航菜单(Suckerfish或其他的菜单)，这都是一个标准过程。
- 技巧2：小心处理IE、Suckerfish和Flash。当IE/Win遇到Flash内容(无论是您创建的广告还是动画)，浏览器会把它置于z-index层叠堆的最上面。这就意味着菜单将位

于Flash内容之下，使用户不可能单击被隐藏的菜单项。解决方法是在Flash对象/嵌入代码中包含一个wmode标签。详请请见http://www.macromedia.com/support/flash/ts/documents/flash_top_layer.htm。

- 技巧3：如果文档有多个层叠层次，要包含一个z-index属性。如果布局中用z-index属性对对象进行分层，Suckerfish导航菜单也必须有一个这样的属性，且位于所有对象之上。可以给导航菜单的父列表元素添加一个z-index属性，(在PGA锦标赛网站)也可以用div容器封装导航菜单，并给div添加一个z-index属性。这样就把导航菜单提升到与之有冲突的对象之上，因此菜单可弹出且位于所有可能接触的对象之上。

3. 底线

既然这样，您可能会问，为什么还要推荐Suckerfish？答案很简单。尽管有这里提到的一些问题，Suckerfish仍是最容易访问的、跨浏览器最友好的下拉菜单解决方案。它也远比任何其他下拉菜单小，更容易更新和维护。如果要在访问量非常大的站点(例如，PGA锦标赛站点每小时的点击次数为数百万)实现这样的菜单，则轻量级的下拉菜单解决方案是首选。

3.3 与Web标准兼容的Flash嵌入

在创建与标准兼容的XHTML标记时，Web开发人员面临的常见问题之一是如何嵌入Flash内容。大多数开发者在发布动画时只是简单复制和粘贴由Flash创建的object/embed标签的标准集。然而它们与各种无效属性和元素一起被加载，因此严重破坏了文档对Web标准的依从性，因为embed是任何W3C规范没有的专有元素。

还好有迂回的解决方案。这里有目前最流行的三种方法来嵌入与标准兼容的Flash内容。

3.3.1 使用Flash Satay方法

Flash Satay方法(http://www.alistapart.com/articles/flashsatay)将删除embed标签和一些object标签中“不必要”的专有属性。这个方法非常管用，但存在一个严重的问题：在IE/Win中，在100%加载之前Flash动画不能被启动。

Satay方法提供了一个迂回的解决方案。该方案通过嵌入一个空的“容器”动画来欺骗IE/Win，并用这个容器剪贴板来加载实际的内容，它的参数(宽度、高度等)与“真正”动画的参数一致。这样，IE就能以流方式成功播放动画了，标记也得到验证，但这是以牺牲开发人员的心智为代价——每一个嵌入动画都需要伴随一个空的容器动画，因此会创建很多额外的垃圾目录，很令人头痛。

3.3.2　用JavaScript编写object/embed标签

在这种情形下，仍保留了object/embed元素但把它们移到了一个外部JavaScript文件中。然后用一系列的document.write(这是JavaScript的一个方法)把Flash内容写入这个文档中。有效性验证软件(W3C在http://validator.w3.org有一个非常优秀的验证软件)看到的只是JavaScript这个有效元素，看不到其中包含的Flash的object/embed代码，因此就成功跳过了object/embed标签。

在PGA锦标赛网站使用的就是这种迂回策略。这不仅使XHTML有效，而且由于使用了JavaScript，就可能完成一些少量的浏览器或插件检测，这要求有可替换(非Flash)的内容。

一旦创建了JavaScript外部文件(因为太长而没有列在这里，在Web浏览器中输入地址http://www.pga.com/pgachampionship/2004/js/flash_home.js可看到其源码)，在XHTML文档中是这样链接的：

```
<script type="text/javascript"
src="http://www.pga.com/pgachampionship/2004/js/flash_home.js"></script>
```

这种方法并不是没有问题。首先，Web标准的追求者会争辩说，JavaScript文件本质上是一个特洛伊木马，它会把无效的、不支持的标记引入XHTML并逃过有效性验证软件的检测(正中要害)。其次，依靠JavaScript输出内容，则要假定用户启用了浏览器的JavaScript(多数人会启用，但为了提高访问速度和避免广告，有些用户会禁用JavaScript)。最后，每一个Flash动画都有自己的外部.JS文件(处理一堆动画不是什么难事，但很快会失控)。

3.3.3　SWFObeject

SWFObejct是在2004版的PGA锦标赛网站几个月之后发布的。SWFObject是目前可用的最敏捷、最健壮的嵌入方法。这个JavaScript模块是由Geoff Stearns创建的，它突破了前面两种方法的限制，提供更简单的标记，这些标记与XHTML 1.0 Transitional及更高版本一

样是有效的。

SWFObject为Flash开发人员提供了所需的一切——播放器检测、为没有插件的内容提供替代内容、提供了通过FlashVars传递额外参数和变量的方法、在指定区域嵌入一个swf的div、甚至一个能旁路播放器检测的变量、无论用户是否有插件都强制显示Flash动画。

SWFObject也是一个非常友好的搜索引擎，在搜索时很少需要Flash内容。用户在文档中可以非常方便地创建一个div并加入通常的HTML文本内容，这些内容可以被搜索引擎检索到，并可为没有安装Flash插件的访问者显示。如果访问者安装了Flash插件，div中的内容就被动画代替。这样安装和没有安装插件的访问者都能很容易地接收到丰富的内容，而Web开发人员不需要太多的工作。

有关更多SWFObject的信息可从http://blog.deconcept.com/swfobject免费下载和使用。

3.4 小结

本章介绍了在Photoshop中创建视觉效果、处理下拉菜单、用CSS设计元素的布局、以及解决XHTML中常见的Flash有效性问题等内容。这里所用的技术会激发大家在使用CSS时进行更深的研究和实验。

下一章我们将研究佛罗里达大学Web网站的重新设计，包括网站的历史、更新遗留内容时所遇到的挑战和所用到CSS标记。

第4章 佛罗里达大学主页UFL.edu

佛罗里达大学(UF)是世界上学术领域最广的公立大学之一，有16个学院。UF可追溯到建立于1853年的东佛罗里达神学院，它在国际教育、科研和服务创立方面有非常悠久的历史，在佛罗里达的所有67个郡都有分支机构。

佛罗里达大学有五万多学生，是美国第二大的大学。仔细浏览整个UF网站，从UF的Web展示可以发现一些趋势。从UF Web站点可以发现Web开发人员(和浏览器开发人员)的关注点有所转移。

本章我们将研究UF采用的有关Web展示的一些策略，以及实现这些策略所用到的技术。

4.1 UF第一个Web站点回顾

UF在1995年发布了一个主页，该主页成为那一时期的典范。页面结构优美自然，从美学角度看有很多亮点。1995年的页面实际上是相当实用的，它的链接在目前UF站点(如图4-1所示)中也能找到。

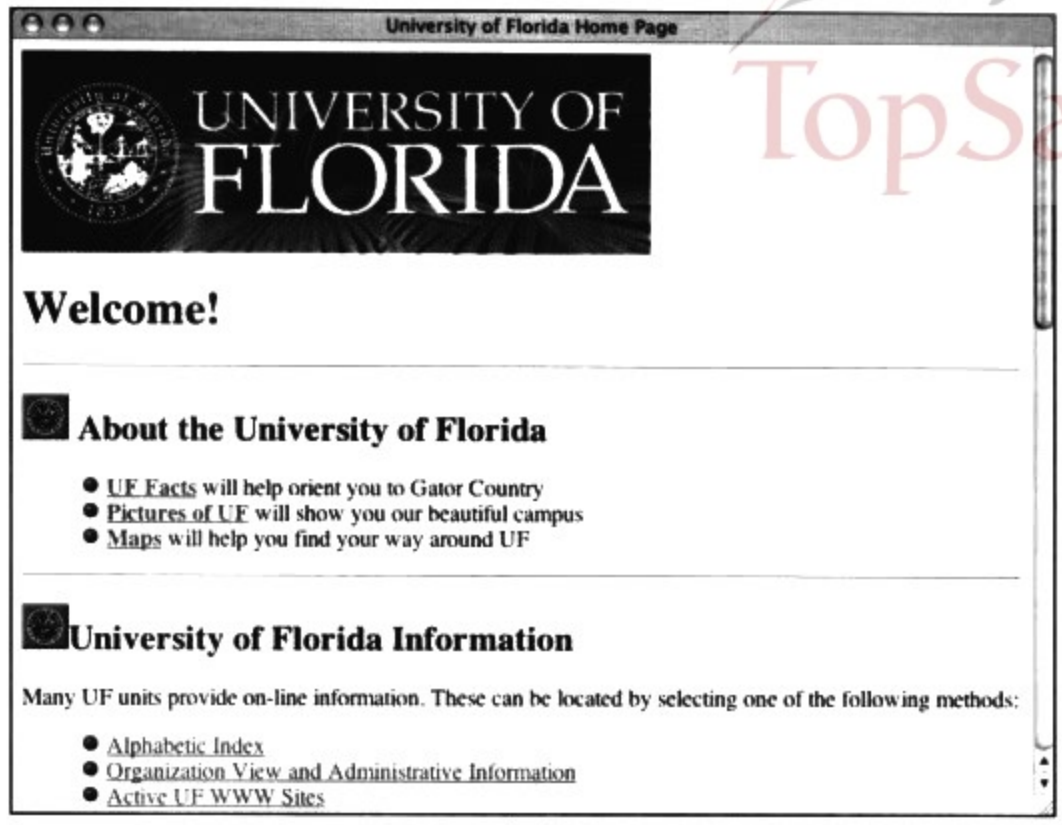


图4-1　1995年发布的UF主页

下面是第一个UF站点的一些标记：

```
<H2><IMG alt=** src="images/placeholder.gif" align=bottom width="32"
height="32">
    About the University of Florida</H2>
    <UL><IMG alt=* src="images/ball.gif" width="14" height="14">
    <B><A href="#">UF Facts</A></B> will help orient you to Gator
Country<BR>
    <IMG alt=* src="images/ball.gif" width="14" height="14">
    <B><A href="#">Pictures of UF</A></B> will show you our beautiful
campus<BR>
    <IMG alt=* src="images/ball.gif" width="14" height="14">
    <B><A href="#">Maps</A></B> will help you find your way around UF<BR>
    </UL>
    <HR>
```

您可能已经注意到，大量的语义元素很容易识别。题头大小合适，无序列表也是这样标注的。列表项元素被明显省略。为了取代li元素，创建者用图像和br标签开始和结束列表项。这样就大致完成了列表的样式设计。

4.1.1　对修改版本的反思

UF网站的后续修改版本逐渐倾向于用基于角色的导航系统。这个系统主要由下列五

个主要的组构成(如图4-2所示)：

- 报考生
- 在校生
- 访问学者
- 教职员工
- 校友、家长和朋友

图4-2　1999年发布的UF主页修改版

在每一组中，访问者能找到每一不同类型的访问者所需的所有信息。“招生”栏提供的是报考信息、校园游览图、大学有关情况等。每一个组都有类似的以角色为目标的信息。基于角色的导航已成为导航的主要手段，在大学的Web网站和其他Web网站都可以看到。

这个以图形为主的新设计并不是没有代价。对于1999年的设计，UF收到很多反馈信息，关于美学方面的正反馈多于负反馈，但在网站加载时间方面有很多抱怨。在这个设计中，访问者的浏览器为一页的不同部分要向UF服务器发三十多个HTTP请求：HTML、图像、JavaScript，等等。每一请求增加了加载网站所需的总时间。

第一个设计中的很多语义明确的标记在修改版中都没有了。第一个设计中的无序列表和题头元素被多重图像映射、JavaScript翻转器和表取代。

4.1.2 对目前网站的分析

随着浏览器在标准兼容性方面逐渐增强，UF决定对网站进行重新设计，以解决上述问题以及其他问题，2004年设计的UF主页如图4-3所示。为了重新设计该网站，UF要完成：

- 大学网站主要用户评估和用户需求分析，包括Web技术的可用性和可访问性需求
- 网站内信息的组织
- 充分的用户测试
- 同类网站比较

目前的网站是在2008年发布的。虽然对设计进行精简了，但核心设计仍十分突出，如图4-4所示。该网站目前每天的访问量大约为十万人次。

本章其余部分将介绍这个新网站是如何开发的。

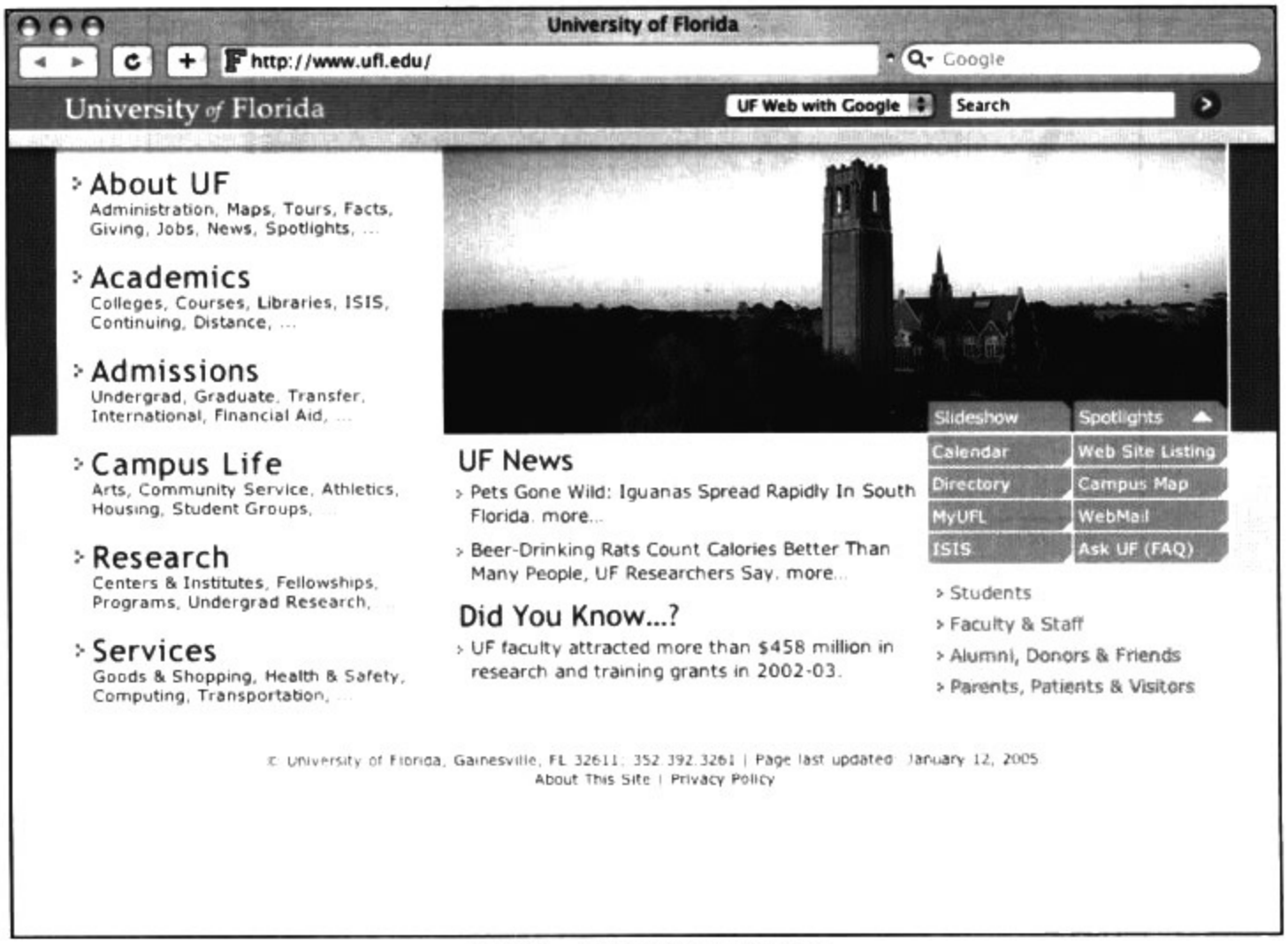


图4-3 2004年发布的网站

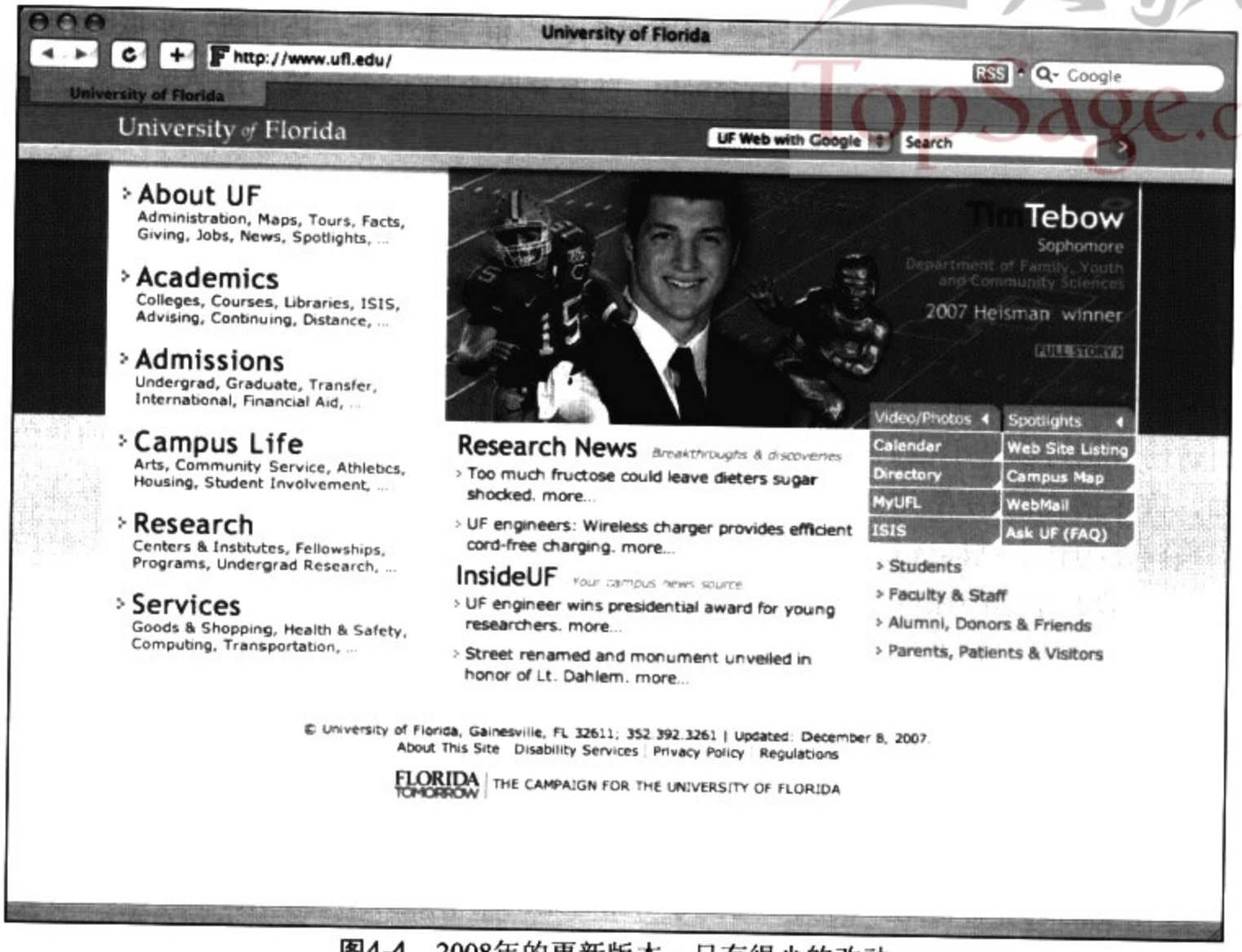


图4-4　2008年的更新版本，只有很少的改动

4.2　网站定义

认识到要使大学网站取得成功，诸如财政开支和购物车等问题是次要的，重要的是访问者的满意度和感觉。要达到这样的目标很简单：与网站访问者交流、看看他们如何使用网站并给他们提供所需要的东西。

4.2.1　组建开发团队

UF团体的许多成员有多重角色。有很多访问者希望在同一网站能搜寻到更多类型的内容。因此，网站设计不仅要由访问者驱动，也不可避免地要由UF教职员工的成员驱动，因为他们直接处理访问者的要求。在早期，团队由校园内二十个领域的代表组成，他们开发了一个嵌套列表来表示访问UF网站的用户类型。例如，主要的用户类型是“学

生”。“学生”又分为“本科生”、“研究生”、“留学生”等。还可进一步嵌套：“考研生”、“转科生”(transfer undergraduate，转专业的本科生)等。然后团队确定每种类型的访问者要完成的任务。

4.2.2 从用户研究着手

和大多数组织一样，UF的人常常有不同的角色。一个教员可能上课也可能参加运动会，一个研究生可能是教师也可能是大学校友。用户研究表明，前一UF网站的访问者必须猜测他们应该属于那一组才能找到他们所需的信息。

访问者也可以基于他们所期望的成员资格选择不同的组，而不必非基于当前的角色。例如，许多潜在的UF学生(也就是中学生)从UF主页选择“在校生”而不是“报考生”链接。当问及为什么这样选择时，有三种回答：

- ·“我是中学的在校生。”
- “什么是‘报考生’？”
- “我希望看看真正的UF学生干些什么，而不是宣传册所宣传的。”(在选择其他链接而不是“报考生”的潜在学生中，这种类型的回答最普遍，而且也最有趣。)

显然，大学对人们进行分类的方法(对大学来讲是有效和有用的)不必和用户对自己进行分类的方法相同。这也表明，基于角色的导航不应该是网站导航的唯一手段。

UF需要把这种看待用户的观点与大学Web网站的当前趋势结合起来。

4.2.3 自我检查

UF网站定义的工作重心是研究整个大学网站是如何定义的。许多大学网站主要采用基于角色的导航系统，而UF开发团队经过用户研究后认为基于角色的导航系统对UF不合适。也就是说，UF开发团队可以借鉴其他大学网站采用的惯用语。大学网站定义不同内容的惯用语，如果适用，可以作为主导航的基础。

UF开发团队一开始制定的目标就是要使UF网站的最后结构独具特色。然而，考察这样的宏观趋势可以得到一些最常用的措辞。为此，UF调查了大量的大学Web网站以搜寻常用的术语。如“学校概况”、“学术研究”、“招生信息”和“校友录”等惯用语迅速浮出水面。这些与用户研究(卡片分类)吻合并被作为导航的主要菜单项。

说明：

在构建信息体系结构时，卡片分类是一种吸引网站访问者的方法，虽然技术含量很

低但非常有效。这种方法的本质是，要求测试参与者对索引卡片进行分组(这些卡片包含站点有关内容和任务的简短描述)，然后对这些新的一叠卡片命名。这个方法的有关技术细节请参考Mike Kuniavsky的*Observing the User Experience: A Practitioner' sGuide to User Research* (Morgan Kaufmann, 2003)一书。

很多网站常采用首字母缩写或其他专用术语。这在某些情况下是不可避免的，但应该避免这种做法，使用时要相当谨慎。UF开发团队也注意到这一点，很少定义首字母缩写。

4.2.4　定义技术规范

在前一设计中，由于大量图像导致下载时间的增加，UF希望在1995年的版本中解决这个问题。UF也希望采用(和改进)最初版本的语义。如果访问者的环境基本支持与标准严格兼容的设计，UF将构建这样的环境。

1. 使用Web标准

在网站开发之初就决定，目标是实现网站内容和美工的分离的。这只有在UF了解下列情况才能实现：

- 如果采用基于标准方案，有多少用户会受到不利影响。
- 这些用户抛弃不兼容浏览器的速度有多快。

UF能自如地继续采用与标准兼容的标记，对XHTML元素的语义用法也相当熟悉，因为：

- Netscape用户可平稳过渡到Netscape7，最后过渡到Mozilla和Firefox。后面我们将做更详细的介绍。
- 一般来讲，不兼容浏览器用户是通过更慢的链接访问Web的。加载时间大大减少(对非兼容浏览器用户，从50KB、30个HTTP请求减到30KB、更重要的是只有3个HTTP请求)远远胜过对美工的简化。对时常受制于对老浏览器反应很慢的页面的用户来说尤其如此。
- 企业资源管理系统弃用老浏览器采用现代浏览器处理大学财务，这也迫使用户转到现代浏览器。(内部学生、院系和员工使用校园网占UF主页访问量的40%。)
- 对于不兼容浏览器的用户，由于浏览器不支持@import规则，即使使用样式表，他们所收到的文档，其样式也是非常简单的浏览器默认样式。

2. 体现可访问性

UF在构建网站时非常关注的是其可访问性，希望UF社区的所有成员能使用各种媒体

访问该网站。包括因为听觉、视觉和肢体残疾而必须采用其他Web浏览技术的群体。基于标准的方案恰恰把增强可访问性作为整个开发过程的一部分，而不是在发布网站时简单地增加插件，或采用更差的做法——在发现不可访问时创建一个“纯文本”的版本。

说明：

关于如屏幕阅读器、盲文显示和语音浏览器等辅助科技的更多信息请参考Joe Clark的*Building Accessible Websites*一书(New Riders Press, 2002)。

4.3 构建主导航结构

图4-5 UF的主导航项

佛罗里达大学有一百万多个Web页面，涉及内容从航空航天工程到动物学。站点导航必须同时体现科研与教学活动的差别，并允许对所有资源进行直观访问。

图形或动态翻转器作为导航工具的固有缺陷是隐藏了基本体系结构。用户不能扫描整个页面内容。用户必须猜测结构的每个分支下隐藏了哪些项，然后根据猜测搜索。

虽然翻转器使用户一次能看到站点结构的一个分支，但是不需要用户猜测就能得到整个站点体系结构概貌会更有用。如图4-5所示的六个部分，在每部分题头附近都显示了相关内容，给出了大学的全貌，这是比较好的做法。

4.3.1 XHTML

UF主页的主导航不仅仅是从一个页面定向到另一个页面的手段，还包含这一页一些最重要的内容，导航所用的标记应该增强这种重要性。

1. 无序列表

按类似的信息分组(如像使用列表时的那样)可以：

- 通过CSS进行样式控制
- 按类似的对象进行语义分组
- 按内容组更容易遍历，特别是对屏幕阅读器

2. 嵌套无序列表

初一看，似乎把UF主导航处理为嵌套的无序列表更好。图4-6是关于主题的一个无序

列表，每个主题都有一组链接。XHTML如下所示：

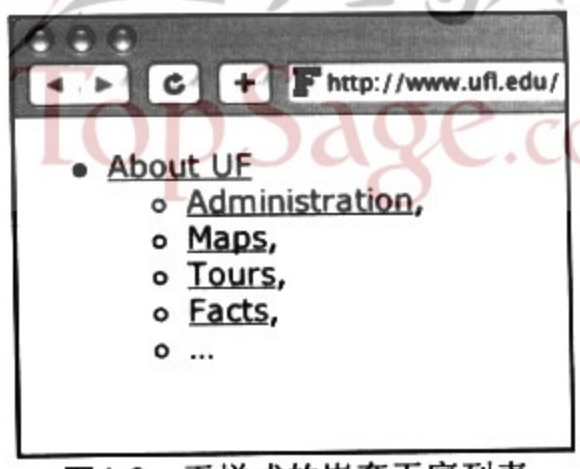


图4-6　无样式的嵌套无序列表

```
<ul id="priNav">
 <li><a href="/aboutUF/">About UF</a>
  <ul>
    <li><a href="/aboutUF/administration.
html">Administration</a>,</li>
    <li><a href="http://campusmap.ufl.
edu/">Maps</a>,</li>
    <li><a href="http://virtualtour.ufl.
edu/">Tours</a>,</li>
   <li><a href="/facts/">Facts</a>,</li>
   ...
  </ul>
 </li>
 <li><a href="/academics/">Academics</a>
  <ul>
   <li><a href="/colleges/">Colleges</a>,</li>
   <li><a href="http://www.reg.ufl.edu/soc/">Courses</a>,</li>
   <li><a href="http://www.uflib.ufl.edu/">Libraries</a>,</li>
    <li><a href="http://www.isis.ufl.edu/">ISIS</a>,</li>
    ...
   </ul>
  </li>
 ...
</ul>
```

虽然有效，但导航内起到题头作用的列表项(About UF、Administration等)与题头元素没有分开定义。当作为列表项对待时，站点的这些主要区域就被淡化了。

3. 加重语义

为了给页面和站点内的主题头赋予总体上正确的含义，导航被放在一个div内，并被表示为一系列跟在无序列表之后的题头，如图4-7所示。

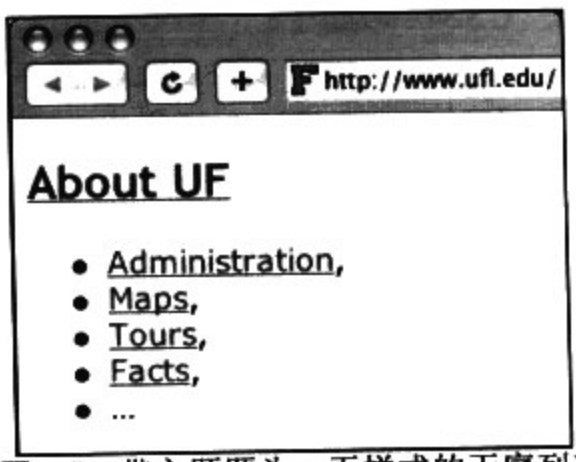


图4-7　带主题题头、无样式的无序列表

```
<div id="priNav">
 <h2><a href="/aboutUF/">About UF</a>
</h2>
  <ul>
    <li><a href="/aboutUF/administration.
html">Administration</a>,</li>
    <li><a href="http://campusmap.ufl.edu/">Maps</a>,</li>
    <li><a href="http://virtualtour.ufl.edu/">Tours</a>,</li>
```

```
      <li><a href="/facts/">Facts</a>,</li>
      ...
    </ul>
    <h2><a href="/academics/">Academics</a></h2>
    <ul>
      <li><a href="/colleges/">Colleges</a>,</li>
      <li><a href="http://www.reg.ufl.edu/soc/">Courses</a>,</li>
      <li><a href="http://www.uflib.ufl.edu/">Libraries</a>,</li>
      <li><a href="http://www.isis.ufl.edu/"><acronym title="Integrated
Student Information System">ISIS</acronym></a>,</li>
      ...
    </ul>
    ...
  </div>
```

这里给题头赋予了当前页面和整个网站的意义。这对屏幕阅读器(如Jaws和IBM的Home Page Reader)的用户很有帮助，可以从题头到题头进行基本的导航。还有一种额外的好处：许多搜寻引擎给题头内的文本赋予了更高的权重。

由于是Integrated Student Information System (UF的一个基于Web的系统，用于注册课程和缴纳学费，在学院的列表项内可看到)的首写字母缩写，ISIS被认为是个缩略语。如果这些首写字母按字母音拼读(就像HTML一样)，这个词被认为是个缩写词并使用abbr标签。Acronym和abbr允许对呈现给用户的文本进行内容扩展。这在文档中第一次出现缩略语或缩写词时进行描述。这帮助屏幕阅读器读出缩略语或缩写词的含义。它也为Google这样的搜索引擎解释了深奥短语的意义。

4.3.2 CSS

CSS代码如下：

```
* {
 padding: 0;
 margin: 0;
}

 ul {
  list-style: none;
}

li {
 font-size: 11px;
 color: #444;
}
```

```
h2 {
 font-weight: normal;
 font-size: 21px;
}

a {
 text-decoration: none;
}

a:link, a:visited {
 color: #0021a5;
}

a:hover, a:active {
 color: #ff4a00;
}

acronym {
 border: 0;
 font-style: normal;
}

#priNav {
 width: 248px;
 float: left;
 padding: 6px 2px 0 0;
 margin: 0 2px;
}

#priNav h2 {
 padding: 7px 0 0 20px;
 letter-spacing: 1px;
 line-height: 22px;
 background: url(images/pointer.gif) no-repeat 9px 14px;
}

#priNav ul {
 padding: 0 0 7px 20px;
 height: 26px;
 border-bottom: 1px solid #eee;
}

#priNav li {
 display: inline;
```

```
  line-height: 13px;
  padding-right: 4px;
  float: left;
}
```

4.3.3 图像

UF用两个图像来显示主导航：navDropShadow.jpg(如图4-8所示)和pointer.gif(如图4-9所示)。第一个是从主HTML元素容器借用过来的。

body元素中的background.gif图像可在图4-3的两边看到。

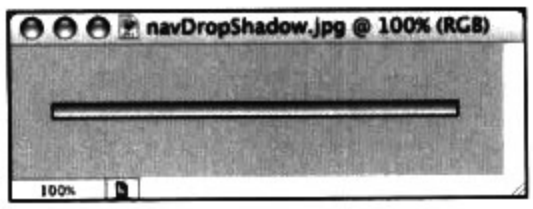


图4-8 navDropShadow.jpg

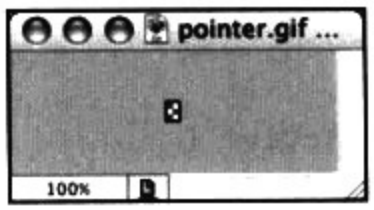


图4-9 pointer.gif.

4.3.4 实现细节

首先，用通配选择符把所有元素的默认外边距和内边距设置为0。外边距和内边距是隔开元素最常用的两个属性。margain指元素之外的空地，padding指元素之内的空地。

```
* {
  padding: 0;
  margin: 0;
}
```

然后删除无序列表前的圆点、加亮h2和li元素：

```
ul {
  list-style: none;
}
li {
  font-size: 11px;
}
h2 {
  font-weight: normal;
  font-size: 21px;
}
```

UF的官方颜色是橘红色和蓝色，因此把所有链接变为UF蓝，把悬停和激活态设置为橘红色：

```
a:link, a:visited {
 color: #0021a5;
}
a:hover, a:active {
 color: #ff4a00;
}
```

UF主页有非常多的链接(ufl.edu有一百万多个页面)，如果该页的每个字都有下划线将非常乱，因此去掉所有链接的下划线：

```
a {
 text-decoration: none;
}
a:link, a:visited {
 color: #0021a5;
}
a:hover, a:active {
 color: #ff4a00;
}
```

站点用了几个acronym元素。一些用户代理使用border-bottom，保留默认的border-bottom将把用户视线转移到有缩略语的链接。这是不希望的结果，因此删除acronym的下划线效果：

```
acronym {
 border: 0;
}
```

4.3.5　边框的构建

主导航有固定宽度，因此首先设置这个宽度：

```
#priNav {
 width: 248px;
}
```

把float属性设置为left以保证导航条出现在左边，其余内容出现在右边：

```
#priNav {
 width: 248px;
 float: left;
}
```

内容容器的左、右内边距为2px，为了与之匹配，把主导航条右外边距也设置为2px。用外边距代替内边距是为了保持离主容器的边有2px、离主内容区2px的可写区域。

```
#priNav {
 width: 248px;
 float: left;
 margin-right: 2px;
}
```

为了给题头阴影一些渐变的空间，并防止文本跑到分隔符的右边，为导航添加上内边距和右内边距：

```
#priNav {
 width: 248px;
 float: left;
 padding: 6px 2px 0 0;
 margin-right: 2px;
}
```

对主导航容器的样式控制就这么多。下面我们介绍段落元素的样式控制。

4.3.6 段落题头样式

主导航内的每个h2涉及UF站点六个主区域中的一个。我们已为h2元素设置了全局性的font属性。在这里为h2添加一个正的letter-spacing，使它稍微稀疏一点：

```
#priNav h2 {
 letter-spacing: 1px;
}
```

主导航容器已把范围设置为段落题头的宽度，因此这里不用再显示设置了。尽管如此，必须为h2的左边和上边留一定的内边距：

```
#priNav h2 {
 padding: 7px 0 0 20px;
 letter-spacing: 1px;
}
```

可以把background-image、background-repeat和background-position属性组合在background属性中。在本例中我们在每个h2元素前放一个圆点(point.gif)。圆点左上角位于刚才设置的内边距区域左上角向右9px、向下14px的位置。也可把background-color和background-attachment组合在background属性里，不过在这里不需要。

```
#priNav h2 {
 padding: 7px 0 0 20px;
 letter-spacing: 1px;
 background: url(images/pointer.gif) no-repeat 9px 14px;
}
```

为了保证后代(如p和y等字母位于基线之下的部分)不影响它们下面的线条，应定义line-height属性：

```
#priNav h2 {
 padding: 7px 0 0 20px;
 letter-spacing: 1px;
 line-height: 22px;
 background: url(images/pointer.gif) no-repeat 9px 14px;
}
```

4.3.7 列表样式

为了控制每个题头下面的无序列表的样式，先把h2元素内边距集的纵向补集添加到ul。h2默认是个块元素。

```
#priNav ul {
 padding: 0 0 7px 20px;
}
```

为了在ul块内一个挨一个地摆放li元素，下一步就要重写li元素的默认块状态，把display属性设置为inline：

```
#priNav li {
 display: inline;
}
```

为了设置列表的高度，为ul元素添加height属性，为li元素添加line-height属性：

```
#priNav ul {
 padding: 0 0 7px 20px;
 height: 26px;
}
#priNav li {
 display: inline;
 line-height: 13px;
}
```

现在添加用于隔离每个部分的边框：

```
#priNav ul {
 padding: 0 0 7px 20px;
 height: 26px;
 border-bottom: 1px solid #eee;
}
#priNav li {
 display: inline;
 line-height: 13px;
}
```

这相当直观，不是吗？这些改变保证了主导航的语义完整，控制了题头和后续列表的样式，使之与期望的设计匹配。

处理了主导航后，再来看看另一种类型的无序列表：辅助导航。

4.4 实现辅助导航

辅助导航功能与其名称吻合。它是主导航的补充，其链接指向导航层次中更低的项，这对用户是有益的。这里讨论的辅助导航由两部分组成：实用工具导航和基于角色导航(如图4-10所示)。

图4-10 UF的辅助导航

注意：

按钮Video/Photos和Spotlights属于Flash动画的一部分，而不是辅助导航。

4.4.1 XHTML

由于辅助导航由截然不同的两个部分组成，因此需要分别定义。容纳两者的结构基本一致。我们先看看实用工具导航。

1. 实用工具导航

实用工具导航仅仅是一组常用工具链接。在UF站点，实用工具导航包括校园地图、活动日程、校园目录、ufl.edu email访问等。在以前，功能不同的组常用横线或图像分隔。既不给这些分隔项添加含义，也不允许浏览器提供功能。

最好的方法是用无序列表对工具类型进行分组，要创建的列表如下所示：

```
<ul id="utilNav">
 <li><a href="http://calendar.ufl.edu/">Calendar</a></li>
```

```
 <li><a href="/websites/">Web Site Listing</a></li>
 <li><a href="http://phonebook.ufl.edu/">Directory</a></li>
 <li><a href="http://campusmap.ufl.edu/">Campus Map</a></li>
 ...
</ul>
```

每个实用工具在无序列表utilNav内都有一个自己的列表项。这就是辅助导航的第一部分，下面处理第二部分。

2. 基于角色导航

前面已提到，在设计中基于角色的导航不是主要的导航手段。尽管如此，UF并不想完全删除这种信息分组手段。

我们可以保留基于角色的导航，但在XHTML中把它移到页面的下面，并置于设计右下角的一个框内，从而降低了它的重要性。

```
<ul id="roleNav">
 <li><a href="/students/">Students</a></li>
 <li><a href="/facstaff/">Faculty &amp; Staff</a></li>
 <li><a href="/friends/">Alumni, Donors &amp; Friends</a></li>
 <li><a href="/visitors/">Parents, Patients &amp; Visitors</a></li>
</ul>
```

与实用工具导航类似，每个角色是无序列表roleNav内的一个列表项。

3. 两者的封装

既然创建了实用工具和基于角色的导航，就可以把它们组合在一起作为辅助导航。只需把一切封装在名为suppNav的div内即可：

```
<div id="suppNav">
 <ul id="utilNav">
  <li><a href="http://calendar.ufl.edu/">Calendar</a></li>
  <li><a href="/websites/">Web Site Listing</a></li>
  <li><a href="http://phonebook.ufl.edu/">Directory</a></li>
  <li><a href="http://campusmap.ufl.edu/">Campus Map</a></li>
  ...
 </ul>
 <ul id="roleNav">
  <li><a href="/students/">Students</a></li>
  <li><a href="/facstaff/">Faculty &amp; Staff</a></li>
  <li><a href="/friends/">Alumni, Donors &amp; Friends</a></li>
  <li><a href="/visitors/">Parents, Patients &amp; Visitors</a></li>
 </ul>
```

```
</div>
```

现在基本上完成了，但还没有对这两个导航组进行定义。我们必须提供标题，这些标题用来描述实用工具和基于角色导航内所包含的内容。

```
<div id="suppNav">
 <h2>Frequently Used Sites</h2>
 <ul id="utilNav">
  <li><a href="http://calendar.ufl.edu/">Calendar</a></li>
  <li><a href="/websites/">Web Site Listing</a></li>
  <li><a href="http://phonebook.ufl.edu/">Directory</a></li>
  <li><a href="http://campusmap.ufl.edu/">Campus Map</a></li>
  ...
 </ul>
 <h2>Information For:</h2>
 <ul id="roleNav">
   <li><a href="/students/">Students</a></li>
   <li><a href="/facstaff/">Faculty &amp; Staff</a></li>
   <li><a href="/friends/">Alumni, Donors &amp; Friends</a></li>
   <li><a href="/visitors/">Parents, Patients &amp; Visitors</a></li>
  </ul>
</div>
```

现在我们拥有一对结构良好的列表，下面需要对它们进行装饰。

4.4.2 CSS

这里有三个截然不同的项：实用工具导航、基于角色导航和辅助导航容器。首先需要处理辅助导航容器。

1. 辅助导航样式

首先要做的是对辅助导航容器suppNav的宽度做一些限制：

```
#suppNav {
 width: 200px;
}
```

必须把整个辅助导航移到正常内容之外，并与页面右边有一定距离。要做到这一点，需要对容器按绝对方式定位：

```
#suppNav {
 width: 200px;
 position: absolute;
 top: 195px;
```

```
  right: 2px;
}
```

已经为两个导航组分别添加了标题，接下来将对每个导航组的样式进行设置。所设置样式应足以把列表与周围信息分开，因此可隐藏辅助导航内的标题：

```
#suppNav h2 {
 display: none;
}
```

因为样式在样式表内，样式表又是通过@import导入的，所以，标题不仅在搜寻引擎而且在老的浏览器中仍能按正确格式显示。

设置了样式后的辅助导航div提供了一个非常好的容器，在这个容器里可以放置实用工具导航和基于角色导航。

2. 实用工具样式

我们希望实用工具列表充满容器的宽度。首先把li元素的display属性设置为inline，这与主导航内的列表设置一致。

```
#utilNav li {
 display: inline;
}
```

然后为每个实用工具创建一个框，实用工具将位于所创建的框内。因为每个框都有准确的宽度和高度，因此需要设置这两个属性。为了利用框内默认的水平居中对齐，需要用line-height属性设置高度。

```
#utilNav li {
 width: 99px;
 line-height: 21px;
 display: inline;
}
```

这个容器的宽度为200px，这给列表项留了一定空间。左边添加的1px外边距应该与底部的1px外边距匹配：

```
#utilNav li {
 width: 99px;
 line-height: 21px;
 margin: 0 0 1px 1px;
 display: inline;
}
```

在确定了列表项的大小和空间后，就可以处理里面的链接了。为了用a元素正确地处理翻转，必须默认地显示类型重写为按块呈现。盛放链接的框大小应与列表项所建的框大小匹配，因此对a元素的大小做相应调整：

```
#utilNav li a {
 width: 99px;
 height: 21px;
 display: block;
}
```

为了给里面的文本留一定的左内边距——且为了避免对box模型不同解释所引起的混乱——采用文本缩进代替内边距：

```
#utilNav li a {
 text-indent: 3px;
 width: 99px;
 height: 21px;
 display: block;
}
```

为了对链接进行修饰，应添加一些颜色和一个图像。把图像放在每个链接的右下角，这样放置把右下角切成45°角：

```
#utilNav li a {
 text-indent: 3px;
 width: 99px;
 height: 21px;
 color: #fff;
 display: block;
background: #94a2d9 url(images/chamfer.gif) no-repeat right bottom;
}
```

给链接添加一个反斜杠的方法与设置悬停状态一样简单。通过改变background-color属性可以对链接的颜色稍做改变，但这个切角图像保持不变。

```
#utilNav li a:hover {
 background-color: #566cc3;
}
```

3. 巧妙的Box模型

使用列表项在IE 5/Mac和IE 5/Win中会出问题。用缩进链接内的文本来代替内边距，解决了除IE 5外的所有浏览器中的呈现问题。当缩进列表项内的文本时，IE 5创建了一个实际外边距，大小等于缩进值，这就使得实际的框大小比期望的宽一些，从而失去了容器内的正确环绕。

要解决这样的怪事也很容易。要使IE 5/Mac和IE 5/Win消除它所产生的额外空间，只需要把所有元素悬浮在左边：

```
#utilNav li {
 width: 99px;
 line-height: 21px;
 margin: 0 0 1px 1px;
 display: inline;
 float: left;
}
#utilNav li a {
 text-indent: 3px;
 width: 99px;
 height: 21px;
 color: #fff;
 display: block;
 background: #94a2d9 url(images/chamfer.gif) no-repeat right bottom;
 float: left;
}
#utilNav li a:hover {
 background-color: #566cc3;
}
```

顺便提一下，在设计工作中完全没必要考虑#utilNav ul元素的样式。为了控制下一导航项的样式，需要用到ul元素。

4. 角色样式

基于角色导航与实用工具导航的样式有一些相似之处，但也有很大区别。我们希望把这种文本效果施加在容器框之内，而不是每个列表项的个别按钮。这种情况下，我们要在无序列表内进行一些处理。

在IE 5/Mac和IE 5/Win中，由于文本缩进造成了额外的宽度，在实用工具导航中为了消除这个额外宽度所采用的方法在这里有其影响。由于实用工具列表项悬浮在左边，因此

浏览器呈现当前元素的点不会移动。解决方法是：清除悬浮在左边的所有元素。

```
ul#roleNav {
 clear: left;
}
```

ul元素应该拉伸到与其容器的宽度一致，因此没有必要设置宽度。高度也是这样自然处理的。为了把基于角色导航的左边与前面创建的单独列表项框的左边排列在一起，应添加1px的左外边距：

```
ul#roleNav {
 clear: left;
 margin-left: 1px;
}
```

实用工具列表项右下角的图像chamfer.gif在这里可重用，把角色列表容器的右下角进行切割：

```
ul#roleNav {
 background: #dbe0f2 url(images/chamfer.gif) no-repeat right bottom;
 clear: left;
 margin-left: 1px;
}
```

列表框需要一些空间进行上下拉伸，因此添加6px：

```
ul#roleNav {
 padding: 6px 0;
 background : #dbe0f2 url(images/chamfer.gif) no-repeat right bottom;
 clear: left;
 margin-left: 1px;
}
```

为了创建列表项之间的间距并把每个列表项置于纵向的中心，需要设置line-height属性：

```
ul#roleNav li {
 line-height: 22px;
}
```

为了消除由IE 5创建的每个列表项之下的额外间距，需要把display属性设置为inline，使之完全占用这部分空间：

```
ul#roleNav li {
```

```
 line-height: 22px;
 display: inline;
}
```

与实用工具列表内的链接一样，需要设置列表项内链接的高度。在这里display属性也应该重写为block。列表项与无序列表一样，这样做将把链接拉伸至容器的宽度：

```
ul#roleNav li a {
 height: 22px;
 display: block;
}
```

为了使文本与边框左边有一定距离，需要添加足够大的左内边距以便放下添加的圆点项目符号：

```
ul#roleNav li a {
 padding-left: 16px;
 height: 22px;
 display: block;
}
```

与主导航中的做法一样，用一个背景图像作为项目符号，这个背景图像位于距左边8px、纵向居中的位置(当background-position的第二个参数省略时，默认为居中)。然后设置字体的颜色和大小：

```
ul#roleNav li a {
 padding-left: 16px;
 font-size: 12px;
 height: 22px;
 display: block;
 color: #596ec4;
 background: #dbe0f2 url(images/pointer_small.gif) no-repeat 8px;
}
ul#roleNav li a:hover {
 color: #0021a5;
}
```

5. 最终结果

现在我们得到了辅助导航的全部样式：

```
#suppNav {
 width  200px
 position  absolute
```

```
    top: 195px;
    right: 2px;
   }
   #suppNav h2 {
    display: none;
   }
   #utilNav li {
    width: 99px;
    line-height: 21px;
    margin: 0 0 1px 1px;
    display: inline;
    float: left;
   }
   #utilNav li a {
    text-indent: 3px;
    width: 99px;
    height: 21px;
    color: #fff;
    display: block;
    float: left; /* For IE5 */
     background: #94a2d9 url(images/whiteCornerNik.gif) no-repeat right
bottom;
   }
   #utilNav li a:hover {
    background-color: #566cc3;
   }
   ul#roleNav {
    padding: 6px 0;
     background : #dbe0f2 url(images/whiteCornerNik.gif) no-repeat right
bottom;
    clear: left;
    margin-left: 1px;
   }
   ul#roleNav li {
    line-height: 22px;
    display: inline;
   }
   ul#roleNav li a {
    padding-left: 16px;
    font-size: 12px;
    height: 22px;
    display: block;
    color: #596ec4;
    background: #dbe0f2 url(images/pointer_small.gif) no-repeat 8px;
   }
```

```
ul#roleNav li a:hover {
 color: #0021a5;
}
```

这就实现了辅助导航。

4.5　再论Flash的嵌入

UF希望分三类翻转显示全体教职员工的spotlight(如图4-11所示)，并决定用Flash(代替Java和JavaScript图像翻转器)来实现。

使用Flash还意味着它不能使交易中断。也就是说，如果用户没有安装Flash，网站仍然要正确地运转。为了适应这部分网站用户，使用Flash会使网站内容增多，这种方法不应给未安装Flash用户的带来新的障碍。

图4-11　UF站点所用的Flash spotlight示例

前一章介绍了在设计中正确包含Flash的方法。现在我们仔细研究其中一个方法：Flash Satay (http://www.alistapart.com/articles/flashsatay)。

4.5.1 Flash Satay方法回顾

在第3章已提到，在Web站点包含Flash的Flash Satay方法使用了embed元素和一些专用属性，老版本浏览器使用embed元素，这些专用属性常用在object元素内。

为了对将要做的事情有更好的了解，先来研究在网页中包含Flash的一段典型标记：

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash
/swflash.cab#version=6,0,0,0" width="516" height="194" id="movie">
 <param name="movie" value="spotlights.swf">
 <embed src="spotlights.swf" width="516" height="194"
  name="movie" type="application/x-shockwave-flash"
  plug inspage="http://www.macromedia.com/go/getflashplayer">
</object>
```

Flash Satay方法去掉了这段标记的一大块，只留下相当简洁的标记：

```
<object type="application/x-shockwave-flash" data="loader.swf"
width="516" height="194">
<param name="movie" value="loader.swf" />
</object>
```

这个新加载器动画(大约4KB的小文件)简单加载含所有内容的更大动画(如图4-12所示)。这就解决了IE/Win中的一个问题：不能按流方式加载object参数中的动画，但这也导致在整个Flash动画加载完毕之前不能播放，用户必须等待。

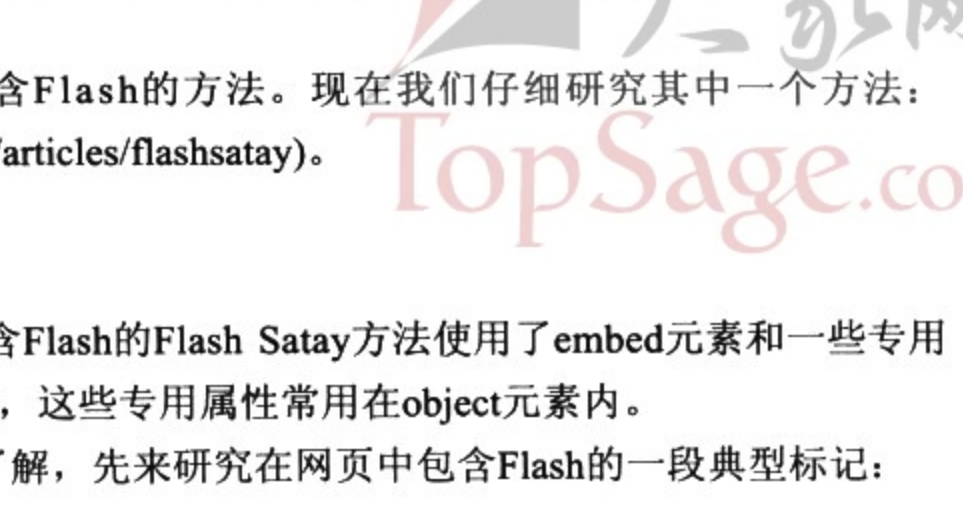


图4-12 Flash Satay使用了一个主Flash内容之外的加载器动画

如果用户没有安装Flash，UF希望使用一个替换内容。object模块有一个非常有用的方法，允许在第一个参数之后简单添加一个替换内容来实现上述要求。在本例所用的是一个校园标志性图像，如图4-13所示。

```
<object type="application/x-shockwave-flash" data="loader.swf"
```

```
width="516" height="194">
    <param name="movie" value="loader.swf" />
    <img src="images/tower.jpg" alt="Century Tower Photo and University
Auditorium" width="516" height="194" />
    </object>
```

在主导航页的其他地方，有一个链接指向相同的spotlight(这些spotlight是基于XHTML的)，基于Flash的spotlight动画也链接到这些spotlight。因此如果用户没有安装Flash或没有启用Flash，仍然能访问到spotlight区的相同内容。

这对所有现代浏览器都有效，或几乎有效。但是存在一个普遍的问题，在用户由IE 5.01升级至IE 5.05时，一个被破坏的Flash Player Active X控件导致所要求的对象(在本例是Flash spotlight加载器)没有classid属性时，呈现的是一个文本区域而不是所要求的对象。要使IE能很好地处理这些与标准兼容的时髦新标记，则需要求助于一些小的“骗术”。

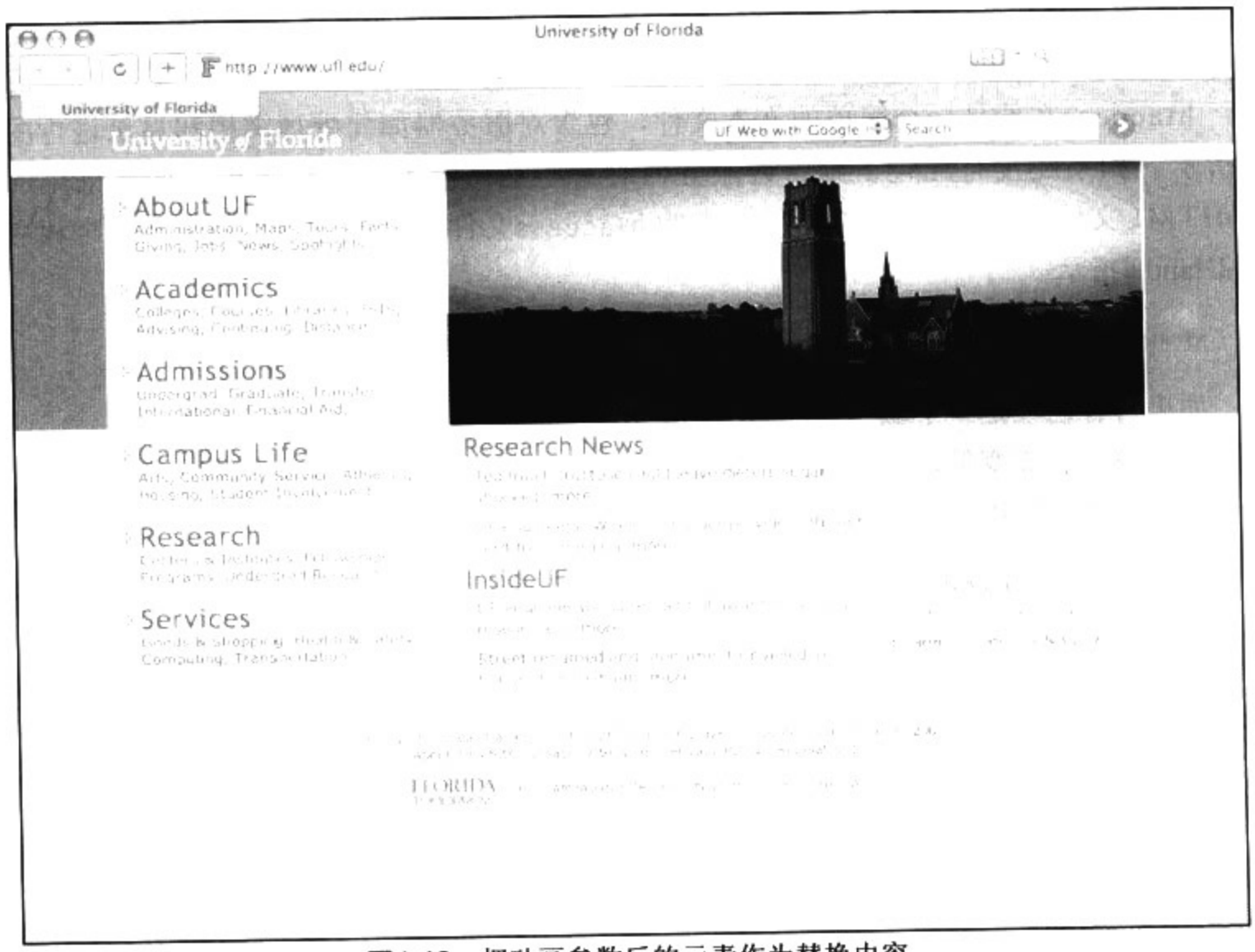


图4-13　把动画参数后的元素作为替换内容

4.5.2 服务器端的Flash Satay检测

Flash Satay建议使用不带embed属性的object元素，把与标准兼容的Flash内容添加到XHTML。UF采用前面介绍的Flash Satay方法来嵌入Flash，并通过小的修改(即服务器端的浏览器检测)解决了IE 5.5中的问题。

1. Apache和BrowserMatch指示

为了正确处理classid属性，仅对使用IE/Win的用户必须传递这个属性。通过浏览器向服务器所做的页面请求可以确定浏览器的类型。服务器通常记录请求有关的基本信息：文件名、时间、请求是否成功、文件大小和浏览器类型(这是我们最需要的)。

这里所用方法假定站点是驻留在Apache Web服务器上的。大约一半的Web站点(包括UF站点)使用的是Apache，这是一个开源Web服务器软件。对于驻留在Microsoft IIS和其他Web服务器上的站点，对该方法做一定修改也可以使用。

首先创建.htaccess文件，这个文件位于站点根目录或使用Flash的文档所在的目录。.htaccess文件是一个简单的文本文件，包含对服务器如何处理文档和目录进行配置的指示。如果还没有配置，则告诉服务器对使用文档的类型进行解析。如果采用的是(X)HTML文档(与本例一样)，则在站点的.htaccess文件中根据文档后缀添加一个相应的AddHandle指示：

```
AddHandler server-parsed .html
```

如果内容已经是一个服务器解析的格式，则不需要添加AddHandle指示。

现在设置环境变量，以便识别传递给IE/Win的文档的内容(也就是classid)。在.htaccess文件中添加如下的行：

```
BrowserMatch MSIE msie
```

这个命令指示服务器，如果用户代理使用的是MSIE(即IE/Win)，则把环境变量msie设置为true。

再研究一下现在的标记：

```
<object type="application/x-shockwave-flash" data="loader.swf" width="516"
height="194 classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000">
<param name="movie" value="loader.swf" />
<img src="images/tower.jpg" alt="Century Tower" width="516" height="194" />
</object>
```

由于有一个变量指明用户使用的是否为IE/Win，因此通过检查msie变量就可以有选择地添加classid。指示服务器这样做的代码如下所示:

```
    <object type="application/x-shockwave-flash" data="loader.swf"
width="516" height="194"
    <!--#if expr="${msie}"-->
    classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    <!--#endif --> >
    <param name="movie" value="loader.swf" />
    <img src="images/tower.jpg" alt="Century Tower" width="516" height="194" />
    </object>
```

有关BrowserMatch指示更详细的介绍请见描述mod_setenvif模块的Apache文档，网址为http://httpd.apache.org/docs/mod/mod_setenvif.html。

2. 服务器端Flash Satay检测的缺陷和障碍

在第3章已经提到，如果一个站点的Flash动画非常多，为每个动画创建容器的工作将非常麻烦。UF站点使用的动画非常少(就一个)，因此很好办。

对于这里介绍的方法，必须对站点服务器进行配置，必须使指示和内容创建者有访问和创建的权限。UF就是这样配置的。

指示必须解析所服务的每一个页面，这给服务器带来额外(虽然是名义上)的开销。这对UF来说不是个问题，因为它已经对每一个页面进行了解析，允许服务器端包含(server-side include，SSI)。

4.6　寻找设计中的失误

经历了七个月的网站重新设计过程之后，UF发现一些地方需要以不同的方法实现。

4.6.1　仅凭示例引导

在UF发布新网站时，校园的院系大声呼吁：要使主页设计中所采用的技术和方式具体化。在权威样式指南没有完成和模板集没在校园内发布的情况下，站点按计划发布了。

随后完成了模板集的开发，但时间上的延误妨碍了校园内网的开发。在最显著的部分经历了这种变化时，加快了UF开发Web站点的管理。

4.6.2　“习惯势力”和“谁移动了我的输入框”

对于任何网站的重新设计，特别是具有大量回头用户的网站，对修改部分进行管理使

之能被接受是非常重要的。经常有这样的用户(特别是员工)，他们对站点的使用大概是这样的："我进入主页，单击链接X，下拉滚动条并单击链接Y，在输入框Z中输入文字，然后单击Go按钮。"

一个经过充分测试的再设计，创建的站点对任何新用户应该是更易于使用的，同时由于巴甫洛夫条件反射，站点的回头用户(如这里引用的)也能迅速认识到站点的变化。

如果对回头用户的迁移缺乏管理和不够重视，将导致大量的反对声和资源的浪费。

在站点与测试用户共享的同时，推迟面向公众的beta版的发布也可以减少一些站点重新设计所带来的痛苦。

4.7 小结

佛罗里达大学Web站点演变给大家展现了一些等待Web站点改变的挑战、使我们了解到UF针对挑战的对策和所采用的技术。我们研究了如何控制不同导航类型中的标题和无序列表的样式，以及为什么要这样用。我们还看到了对广泛采用的Flash标记的一些修改，从而满足与标准兼容的和跨平台的应用。希望这对您的下一个大型项目有所帮助。

第5章 Stuff and Nonsense：CSS切换策略

在这一章，希望您忘掉迄今为止所学的有关CSS的一切，而要透过事物的表面看到本质……或类似的东西。

通过前四章的学习，值得记住的是层叠样式表不仅对站点设计人员很方便，还极大地改善了站点用户的网络生活现状。在第1章简要介绍了如何根据用户头脑中的需求编写层叠样式表，我们也知道用户样式表的优先级比作者编写的样式表高。规范制定者这样做不会妨碍网站设计人员，但给阅读规范的人员授予了相当的权力。

总之，Web的真正奇异之处是对统一访问的承诺：通过这种途径，世界任何地方的用户可以获得任一主题的即时完整的路径。事实上，一个Web设计人员应尽量遵守这一承诺——使站点立即引起人们视觉上的注目且呈现无障碍路径的界面。

然而，设计人员慢慢认识到他们对公众要求的理解是不全面的。在Web的早期，开发工作重点考虑在现代桌面浏览器上如何使网站“看起来是正确的”。但在近期，对用户需求的理解逐渐成熟。有肢体、听觉、视觉或认知残疾的用户均在使用Web站点，只不过设计人员要认识到这一点需要一定的时间。因此只是在最近，可访问性定义才成熟了，站点构建技术也日趋成熟。

一些设计人员可能会告诉您，构建一个可访问的站点意味着构建一个令人厌烦的站点，然而可访问性并不是采用更大的字体和创建高对比度的方针就能实现。一些Web用户只能浏览小字体文本，而另外一些则只能在黑色背景下浏览黄色文本。本书所研究的许多

设计技术(语义、良构标记、内容和表现形式的分离)在令人鼓舞的专业设计中将起到难以置信的杠杆作用，同时会改进所有用户而不是特定几类用户对站点的可访问性。也就是说，我们能更好地了解统一访问的Web潜力，开发一些极好的站点。

本章不是关于可访问性、空间分配的声明，也不会一味强调贫乏的技术会成为最大的障碍。相反，我们将研究不同的技术，使用户利用样式表切换可对设计进行更改。通过对标记文档应用不同的CSS文件，可以对设计的某个或所有方面进行修改，如布局、版式或调色板。这一技术可能引起设计人员的极大兴趣，因为这种技术将大大降低一个站点重新设计的成本。这种技术同时还能给站点用户带来很多好处，允许他们对站点页面进行更小粒度的控制，使访问站点内容更容易。总之，这种技术会使站点设计和访问尽可能的容易。

5.1 基础准备

和其他章一样，我们先介绍一个有效的XHTML文档。作为样式表切换实验，程序清单5-1的文档能很好地达到这个目的。

程序清单5-1：样式切换器实验的标记基础

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>

<title>Always offer an alternative.</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

</head>

<body>

<div id="container">
  <div id="content">
    <h1>Always offer an alternative.</h1>

   <p><span class="lead">Lorem ipsum dolor sit amet,</span> consectetuer
adipiscing elit. Nullam tortor. Integer eros...</p>
```

```
        <p id="blurb">This is, as they say, a &#8220;pull quote.&#8221;</p>

        <p>Donec id nisl...</p>

        <h2>Additionally, you might consider...</h2>

        <p><img src="portrait.png" alt="An author's handsome (if pixellated)
mug" class="portrait" /> Quisque sit amet justo. Cum sociis...</p>
      </div>
    </div>

    </body>
    </html>
```

到现在为止，这种标记对我们似乎没什么新颖的。这个标记很简单，意思也很明了，在文档的大纲(不得不承认没什么意义)位置应用了正确的标题元素(h1和h2)。用<p>元素标记段落，其中有一个id为“blurb”的标记，因此我们在后面可对它采用与其兄弟不同的样式。为了使布局稍微丰富一点，还包含了本书一个作者的照片。

图5-1显示了这个标记的效果，不过现在还比较简单。

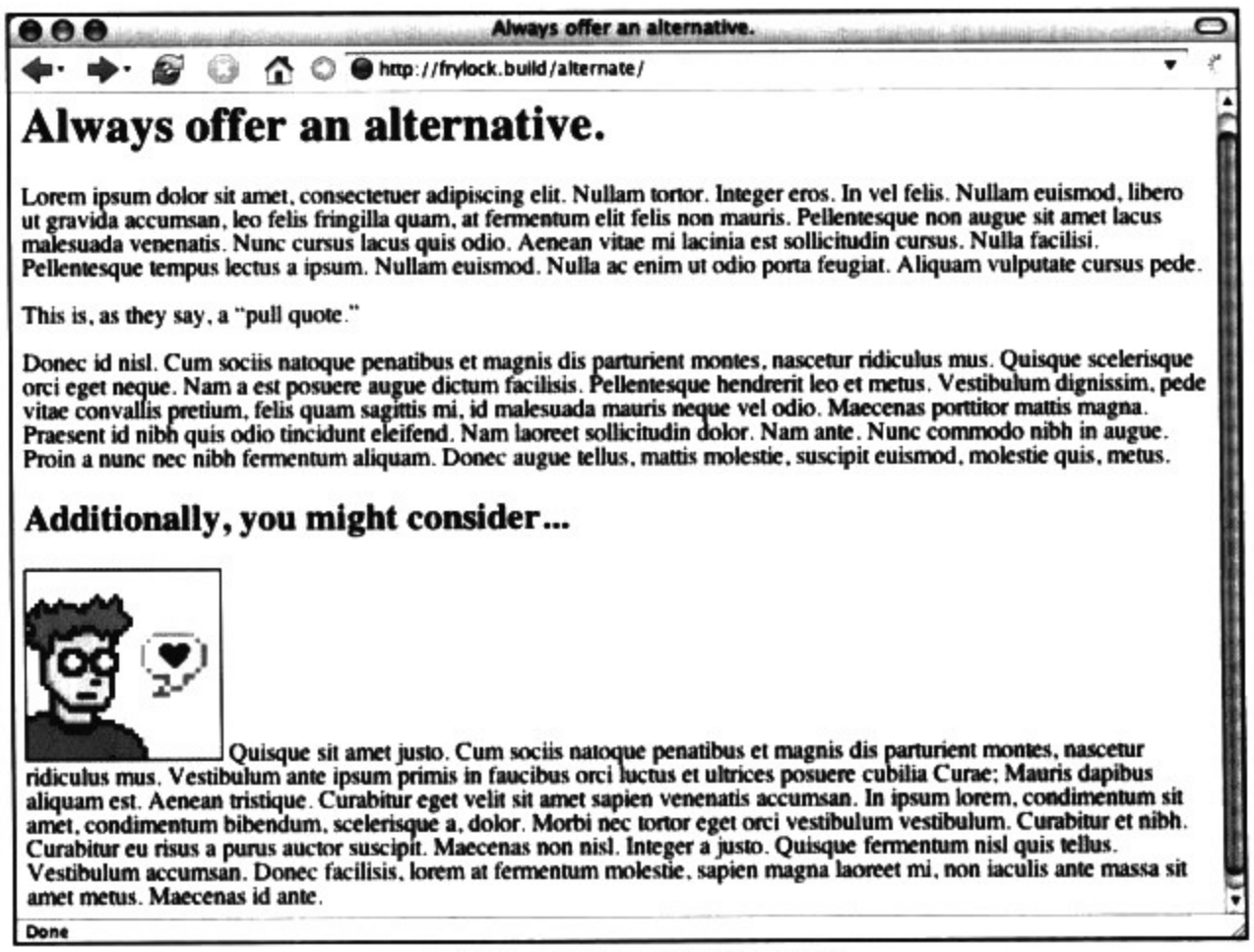


图5-1 无样式XHTML文档

和往常一样，我们把一个简单样式表应用于这个文档，并对样式表稍做修饰。因此创建一个新的样式表core.css，并包含程序清单5-2所示的规则。

程序清单5-2：core.css样式表

```
/* default font and color information */
body {
 background: #FFF;
 color: #444;
  font: 62.5%/1.6em "Lucida Grande", Verdana, Geneva, Helvetica, Arial,
sans-serif;
}
/* END default font and color information */

/* default link rules */
a {
  color: #C60;
}

a:hover {
  color: #F60;
  text-decoration: none;
}
/* END default link rules */

/* headings */
h1, h2 {
  color: #B61;
  line-height: 1em;
  font-weight: normal;
  font-family: Helvetica, Arial, Geneva, Verdana, sans-serif
  margin: 1em 0;
  padding: 0;
}

h1 {
  font-size: 2.2em;
}
/* END headings */

/* container */
#container {
  margin: 0 auto;
  max-width: 60em;
}
```

```
/* END container */

/* content */
#content h2 {
  font-size: 1.2em;
  text-transform: uppercase;
}

#content p {
  font-size: 1.1em;
  line-height: 1.6em;
}

#content img.portrait {
  float: right;
  margin: 0 0 1em 1em;
}

#content span.lead {
  text-transform: uppercase;
}

#content #blurb {
  background: #FFC;
  border: 1px dotted #FC6;
  color: #000;
  font-size: 1.5em;
  line-height: 1.4em;
  padding: .5em;
  text-align: center;
}
/* END content */
```

除了一次添加几个选择符和讨论其视觉结果外，本章更关心切换后的最终效果，而不是样式表所包含的技术。毕竟这个CSS仅仅是一个占位符，可以很容易将其替换掉。也就是说，我们不必深究文件的细节，虽然有以下这些技术值得关注：

- #container { margin: 0 auto; }——虽然您可能对使用没有实际语义价值的div持疑义，但#container是控制其内容宽度的便捷手段。把左边和右边设置为“自动”外边距(0 auto；)可以与父元素(即body标记)内的div水平居中对齐。

  这个方案只有一个小问题。IE 5.x/Win不能正确实现自动外边距，因此它们不能理解这个规则。因此需要对body元素应用text-align: center;。必须承认，这是对text-align属性的不正确解释设计这个属性的目的是用来控制块内的内联内容的，

而不是块本身。但是它能使IE 5/Win满足我们的要求。但是，对body应用{text-align: center;}后，IE 5/Win把页面所有文本都居中对齐了。值得庆幸的是，一旦设置了#container { text-align: left; }，就解决了这个问题。

- #container { max-width: 60em; }——设置max-width属性的效果就是保证#container元素绝不会超过60em。如果用户减小窗口的宽度且小于#container的宽度，div会根据情况缩小。这是一个没有冲突的方法且能获得真正灵活的布局。

  max-width属性最大的缺陷是IE完全不支持。不论是IE/Mac还是IE/Win都不能理解max-width。因此我们必须为IE提供一个已定义的宽度。在本例中定义的宽度是60em，但我们可以选择更灵活的百分比宽度。把这个值提供给IE，通过使用一个智能的CSS hack或在单独的样式表(含与浏览器相关的hack)中放一个“不正确”的值，IE就可自己确定宽度。在第2章已详细介绍了这两种方法。

- body { font: 62.5%/1.6em “Lucida Grande”, Verdana, Geneva, Helvetica, Arial, sansserif;}——在body元素中设置的字体属性实际上是一个速记属性。在简洁的属性/值对中，它声明了font-size (62.5%)、 line-height (1.6em)和font-family (Lucida Grande, Verdana,Geneva, Helvetica, Arial, sans-serif;)。由于这些值被body的后代元素继承，因此，用这一规则可以立即把一个基本的字体配置应用到整个文档。

  font-size为62.5％是由Web设计人员Richard Rutter (*How to Size Text Using ems*, http://clagnut.com/blog/348)首次提出的技术。由于所有现代浏览器的默认文本大小为16px，把font-size设置为62.5％后，body得到的默认字体高度为10px(16 × 0.625 = 10)。把后代元素的字体大小单位设置为em更符合逻辑：1em是10px，1.6em就是16px，0.9em就是9px，等等。在整个文档中不用像素(如果用像素，IE用户不能调整大小)控制，在字体设置时该方法提供了与像素类似，并且用户能调整文本大小，使之满足需要。

- #content img.portrait { float: right; }——这个非常漂亮的像素肖像与包含它的段落右对齐。但是我们没有用如那样受抨击的标记技术，而是用强大的CSS float模型达到了同样的效果。(在此给大家推荐CSS大师Eric Meyer的优秀文章："*Containing Floats*，网址 http://complexspiral.com/publications/containing-floats。)

我们可以很容易把整个这段代码粘贴到文档头的<style type="text/css">...</style>块中，当然这很容易使我们把标记与表现形式的信息搞混——老实说，谁希望在几百个XHTML文档中去搜寻样式信息呢？非常正确，一些人称之为“坚持结构与样式的严格分离”。也可以把这称为“懒惰”。无论您怎么想，保持结构与样式分离会使站点相关工作

变得非常容易处理。

因此，创建第二个样式表，并将其命名为main.css。在这个文件的头部用@import命令导入core.css文件：

```
@import url("core.css");
```

这实际上创建了一个封装样式表——也就是一个CSS文件，它是导入其他样式表的网关。有了main.css文件，就可以在XHTML文档的头部用一个简单的link元素包含它，同时也包含了core.css文件：

```
<link rel="stylesheet" href="main.css" type="text/css" />
```

一个看起来相当普通的文档突然变得有点个性了(如图5-2所示)。

在创建了这个似乎是多余的main.css文件后，您可能直挠头，但确实有非常好的理由要这样做。虽然设计目的不是这样的，但@import规则确实是对浏览器是否支持高级CSS技术的很好测试。一些违反CSS实现的老浏览器不理解@import规则并简单地把它忽略。这样就可以给现代浏览器提供极新颖的样式表规则。而如NetScape 4.0以下版本和IE这样的陈旧浏览器就会简单忽略掉它们不理解的样式表。

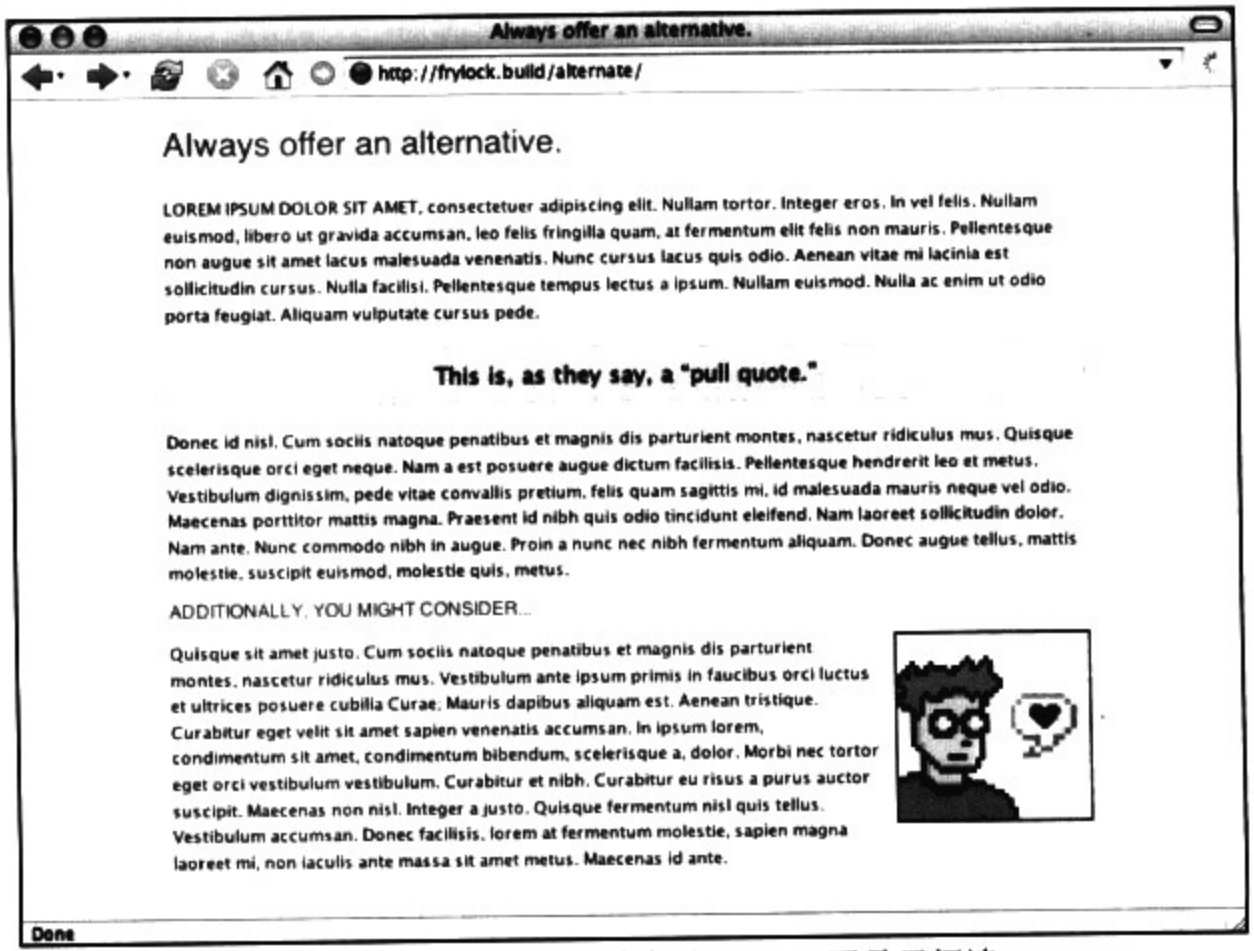


图5-2　运用一些基本样式后XHTML更易于阅读

使用中间样式表的另一个好处是可以在里面添加多个@import规则。在需要把表现形式拆分成多个文件时(如一个用于布局，另一个用于颜色，另一个用于版式)，这就会很方便。甚至更好的是，可以用这个技术管理不同的 CSS hack，就像我们在第2章看到的那样。在测试程序清单5-3的CSS时，可能会发现不同版本的IE(包括Macintosh和Windows平台)违反了布局的不同方面。我们可在core.css文件内用一些备用的样式表hack，为浏览器违反的CSS实现提供一些可供选择的属性值。程序清单5-3说明了如何用封装样式表导入与浏览器有关的CSS hack文件，使main.css文件保持干净整洁且无hack。

程序清单5-3：修改后的含智能Hack管理的core.css文件

```
@import url("core.css");

/* Import WinIEx-only bugs - hide from Mac IE5 \*/
@import url("hacks.win.iex.css");
/* END hide from Mac IE5 */

/* Import Win IE5x hacks */
@media tty {
    i{content:"\";/*" "*/}} @import 'hacks.win.ie5.css'; /*";}
}/* */

/* Import Mac IE5 hacks */
/*\*//*/
@import url("hacks.mac.ie5.css");
/**/
```

我们已经研究了在不理解CSS的浏览器中能正确显示页面所需的CSS hack，也知道把不同hack引入与浏览器有关的文件中的方法，这是保证core.css文件干净整洁和无hack的很好的方法。如果您决定不支持某些浏览器器，只需在main.css文件中删除几行即可，与在原来的样式表规则中一行行地查找CSS hack相比，这确实是一个更有吸引力的想法。在CSS Best Practices Headquarters再次声明，这有一定的“战略性”并有一些“懒惰”成分。

在了解CSS和XHTML后，下面将研究样式表切换的机制。

5.2 CSS切换

目前的示例页面在设计上有一个缺陷，即没有把易读性作为我们的主要设计目标。对比度有点弱，因为文本颜色是与黑色相近的颜色而body的背景为白色。默认字体大小为浏览器的默认值62.5%(大约为10px)，这对有视力残疾的用户来讲就很难阅读(即使轻度近视

的用户也较难阅读)。在最初版本的基础上，如何改进设计使之更易读呢？

我们先创建一个独立的样式表以解决一些可能的陷阱。程序清单5-4提供了一个新的样式表，将其命名为contrast.css。

程序清单5-4：contrast.css样式表

```
body {
  background: #000;
  color: #DDD;
}

h1, h2 {
  color: #FFF;
  font-weight: bold;
}

#content {
  font-size: 1.1em;
}

#content h2 {
  font-size: 1.6em;
  text-transform: none;
}

#content #blurb {
  background: #222;
  border-color: #444;
  color: #FF9;
}

span.lead {
  font-weight: bold;
}
```

现在在标记的头部简单添加一个指向contrast.css文件的链接，如下所示：

```
<link rel="stylesheet" href="main.css" type="text/css" />
<link rel="stylesheet" href="contrast.css" type="text/css" />
```

在浏览器中重新加载这个文档时，就会发现有相当大的变化，如图5-3所示。

首先值得一提的是，由于两个CSS文件一前一后地包含进来，在main.css文件中声明过的布局或字体，在contract.css文件中不需要再做任何声明。更好的方法是，可以有选择地重写个别规则和/或属性值，使这些优先级高的规则对那些需要向用户层叠的规则进行

处理。

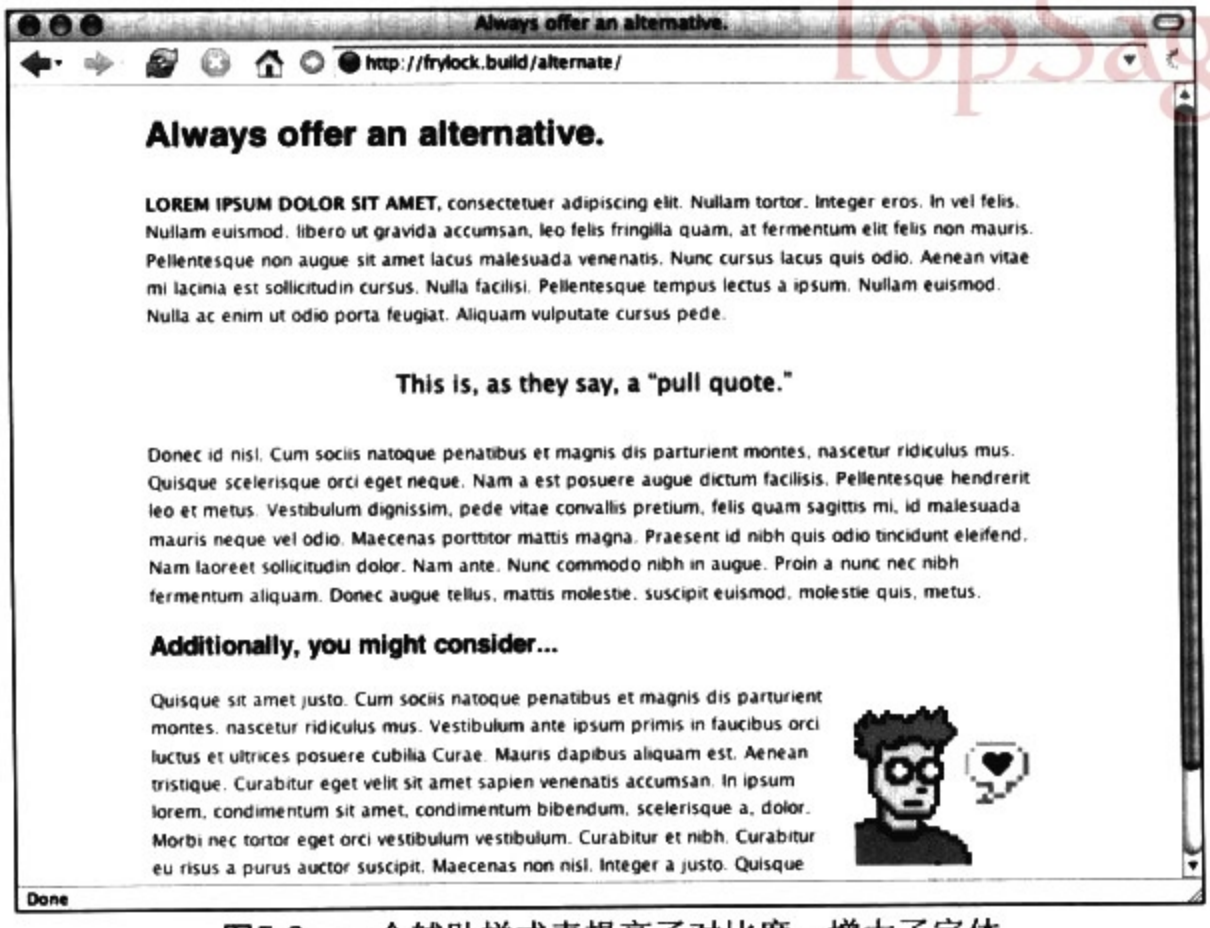


图5-3　一个辅助样式表提高了对比度、增大了字体

从纯美学角度，我们可以即时改变标记的表现形式——通过包含新的contrast.css文件。文本大小的增加非常小(从1em增加到1.1em)，改变引用文字的颜色以反映新的调色板。修改并不大，但重要的是现在用户感到对比度更高，字体大小更易于阅读。

5.3　工作机制

迄今为止，我们已把两个独立的CSS文件合并到一个文件。目前，两个文件以相同权重读入并应用到这个文档，优先级规则解决了这两个CSS文件可能引起的冲突。为了适应更复杂的场景，HTML和CSS规范为多个样式表如何相互作用提供了结构上的指南。Web页面的作者有很多方法可以确定用link元素导入的样式表的优先级。下面我们研究三种不同的样式表——永久的、首选的和备选的，并看看它们是如何应用到切换场景的。

5.3.1　永久样式表

永久样式表总是启用的。可以认为它们是默认为“启用”的CSS。永久CSS文件可以

应用于任何其他样式表，只要这些样式表当前是激活的。永久CSS文件还相当于一个共享样式规则集，文档中的任何其他样式表都可以引用。

带rel属性且被置为stylesheet的每个link元素就是一个永久样式表，事实上我们已创建了两个：

```
<link rel="stylesheet" href="main.css" type="text/css" />
<link rel="stylesheet" href="contrast.css" type="text/css" />
```

在添加其他类型的样式表时，被设计为永久的任何链接都起到一个基线的作用，和其他被包含的CSS文件一样，都可共享它们的规则。

5.3.2 首选样式表

通过给永久样式表添加一个标题，就得到了首选样式表，如下所示：

```
<link rel="stylesheet" href="main.css" type="text/css" />
<link rel="stylesheet" title="Higher Contrast" href="contrast.css"
type="text/css"/>
```

另外，根据title属性可以把首选样式表分成多个“组”。分组后用户就可以同时把这些CSS文件组激活(或关闭)。要呈现多个组时，第一组的优先级最高。

与永久样式表一样，首选CSS文件默认也是启用的。因此在前一个示例中，在用户首次访问该页面时，contrast.css文件就被启用(和以前的做法一样，从永久性的main.css文件中启用)。然而，如果用户选择一个备选的样式表，首选样式表就会被关闭。

5.3.3 备选样式表

备选样式表是首选CSS文件(由站点作者标记)的替代者，它可由用户选择。要把一个链接指定为备选样式表，它必须有title属性且其rel属性被设置为alternate stylesheet。和首选样式表一样，通过title属性可以把备选样式表分组。这正是我们所希望的，通过这种手段，用户可以选择最适合他们需求的设计。事实上，如果希望把main.css设置为默认的，而把contrast.css设置为可选的备选CSS文件，可以更新这两个link元素，如下所示：

```
<link rel="stylesheet" href="main.css" type="text/css" />
<link rel="alternate stylesheet" title="Higher Contrast" href="contrast.
css" type="text/css" />
```

在支持样式表切换的浏览器中浏览该页面，现在用户可以控制该页面的显示。像Firefox和Opera这样的浏览器包含一个选项，通过这个选项可以选择新的备选样式表，如

图5-4所示。

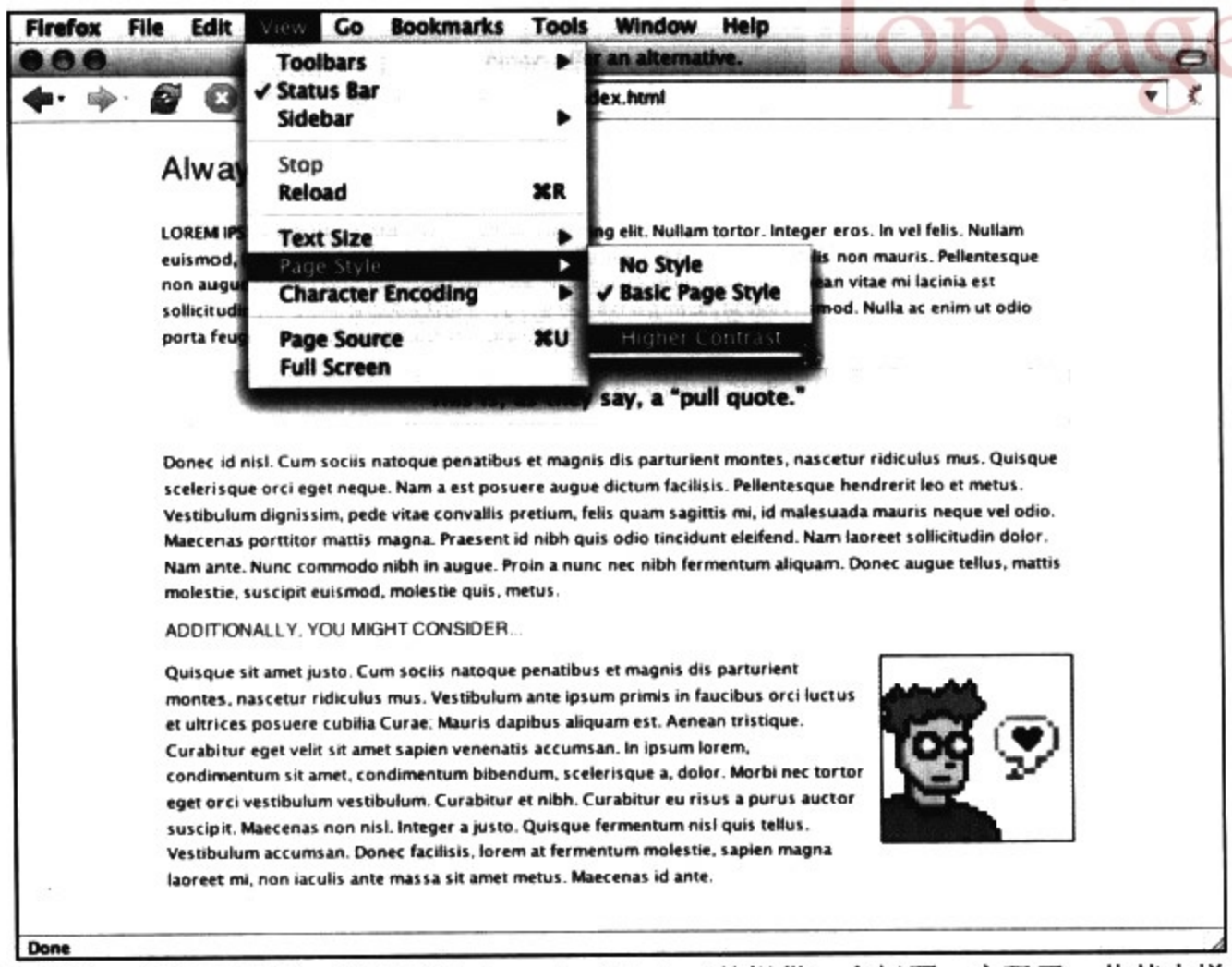


图5-4 把第二个link元素的rel属性改为alternate stylesheet并提供一个标题，实现了一些基本样式切换

一旦用户从菜单选择了Higher Contrast，具有这个标题的样式表(即contrast.css)就被激活。这就是我们寻找的解决方案。最初的设计是默认激活的，但我们提供了一种手段供用户选择另一个样式表。利用这种方法可以添加更多的备选CSS选项。例如，创建一个hot.css文件并使用程序清单5-5中的规则。

程序清单5-5：hot.css样式表

```
body {
  background: #000 url("bg-stylish.jpg") no-repeat 50% 0;
  color: #DDD;
}

h1, h2 {
  color: #FFF;
  font-weight: normal;
  text-align: center;
  text-transform: none;
```

```
    }

    #content {
      font-size: 1.1em;
    }

    #content h1 {
       font: 2.6em Zapfino, "Gill Sans", Gill, Palatino, "Times New Roman",
Times,serif;
       margin: 200px 0 70px;
    }

    #content h2 {
       font: 1.6em "Gill Sans", Gill, Palatino, "Times New Roman", Times,
serif;
       margin: 1.4em 0;
       text-transform: uppercase;
    }

    #content #blurb {
      background: #222;
      border-color: #444;
      color: #FF9;
    }

    span.lead {
      font-weight: bold;
    }
```

现在应用我们所学的有关备选样式表知识，可以很容易把hot.css文件作为另一个用户界面选项呈现给用户。

```
    <link rel="stylesheet" href="main.css" type="text/css" />
    <link rel="alternate stylesheet" title="Higher Contrast" href="contrast.
css" type="text/css" />
    <link rel="alternate stylesheet" title="Gratuitous CSS" href="hot.css"
    type="text/css" />
```

如果用户能从浏览器中选择一个备选CSS文件，他们就能看到新的样式，如图5-5所示。和以前一样，这个变化相当大，但最终使用户能选择感兴趣的设计并能对内容进行裁剪以满足他们的需求。

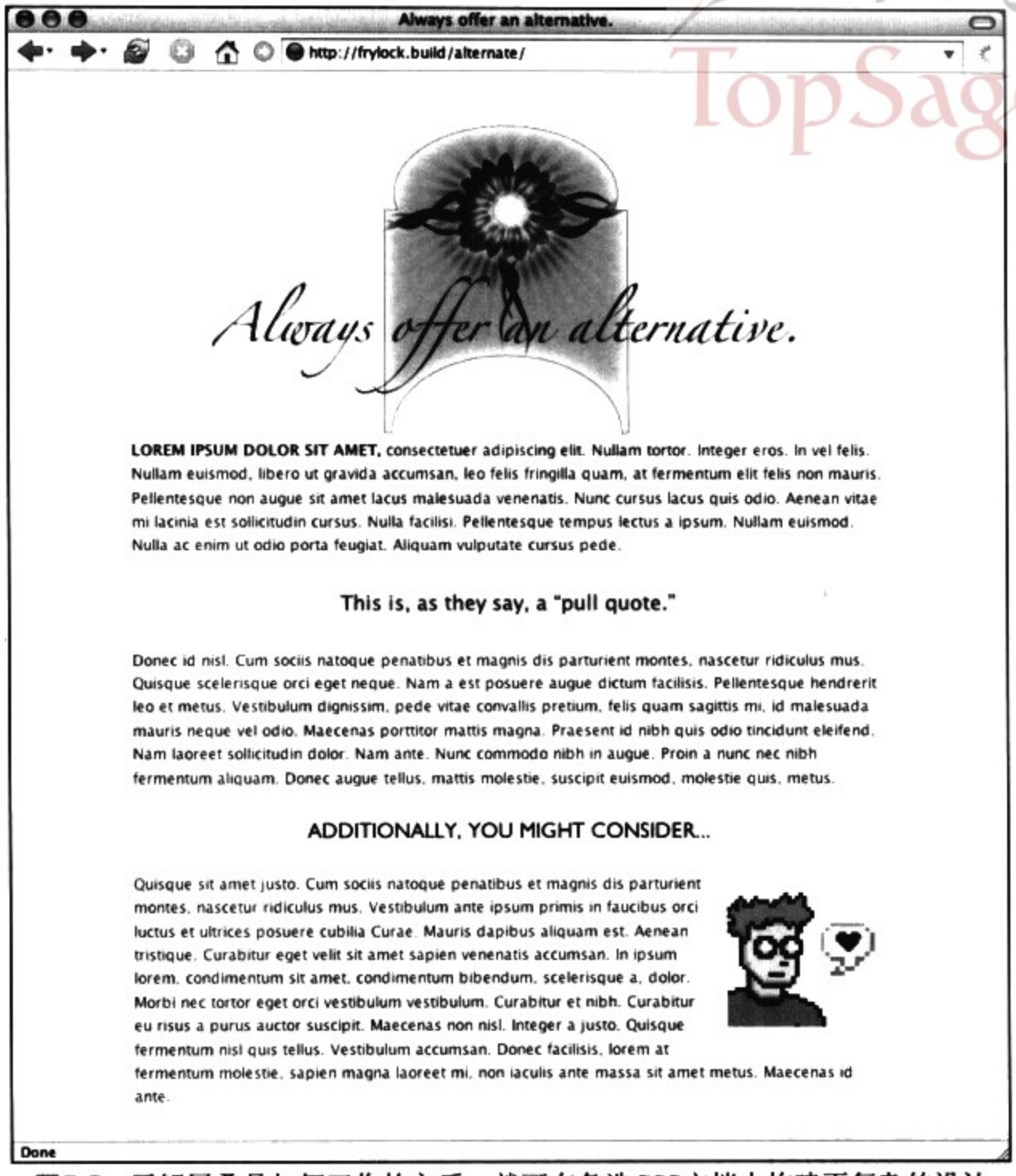


图5-5 了解层叠是如何工作的之后，就可在备选CSS文档中构建更复杂的设计

5.3.4 又一个(几乎)完全不能用的方案

和一些CSS最吸引人的特色一样，如果浏览器能更强地支持备选样式表，它将得到更广泛的采用。在撰写本书之时，自身就允许用户选择备选样式表的浏览器仅限于基于Gecko的浏览器，如Mozilla、Firefox和Opera。例如，Apple的Safari就没有办法选择备选或首选样式表。您可能已猜到，IE(世界知名的和深受喜爱的浏览器)就不允许用户选择网站设计人员创建的备选用户界面。如果世界上最流行的浏览器都不支持这种方案，我们就有

很多工作要做。

进一步讲，本身就支持备选样式表切换的浏览器也仅限于功能性切换。这些浏览器允许用户在默认CSS和作者提供的任何备选CSS之间非常容易地进行切换，但它们不能记住用户的选择。这意味着，如果用户选择了一个备选样式表，然后重新加载该页面或者离开后又回到该页面，浏览器将忘记以前的选择并恢复为默认样式。

很显然，这两种情况都不能使用户满意。幸运的是，可以采取一些额外的步骤使用户能完全享受到CSS切换带来的好处。

5.4 目前的解决方案

虽然创建了CSS切换，但大多数用户在浏览器中不能使用。因此必须在页面中构建一种接口，使用户能克服这些限制。现在您可能认识到我们研究的两种客户端技术在处理上述问题时并不是特别有效。而XHTML和CSS分别在内容描述和样式控制方面非常有效，但它们都不是为用户交互而设计的。当然可以用XHTML在页面内构建如下所示的链接表：

```
<div id="switcher">
    <ul>
     <li id="style-default"><a href="styleswitch.html">Default style</a></li>
     <li  id="style-contrast"><a  href="styleswitch.html">Higher  Contrast
</a></li>
    <li id="style-hot"><a href="styleswitch.html">Gratuitous CSS</a></li>
    </ul>
    </div>
```

还可以在core.css文件中添加一些CSS来控制它们的样式，效果如图5-6所示。

```
/* switcher styles */
#switcher ul {
  text-align: right;
  list-style: none;
}

#switcher ul li {
  border-left: 1px solid;
  list-style: none;
  display: inline;
  padding: 0 0 0 1em;
  margin: 0 1em 0 0;
```

```
}

#switcher #style-default {
  border-left: 0;
  padding-left: 0;
}

#switcher ul a.now {
  color: #000;
  font-weight: bold;
  text-decoration: none;
}
/* END switcher styles */
```

图5-6　一个切换器的链接被添加到页面的顶部，此刻它们看起来很不错

然而当用户单击这些链接时会发生什么？如果您的回答类似于“没什么”，那么您的回答完全正确。当我们讨论对用户的动作有什么响应时，XHTML和CSS不能有任何真正的作用。它们能分别影响页面的内容和表现形式，但当用户试图单击某个链接来更改激活的样式表时，就需要求助于第三方工具：JavaScript。

5.4.1　转向JavaScript

简而言之，JavaScript是一种客户端的脚本语言。JavaScript(或简写为JS)是一种在Web

页面添加一个交互层的语言。当用户访问一个含JavaScript代码的页面时，浏览器就读取这段JavaScript并执行可能包含的指令。这些指令告诉浏览器，在用户完成一个表格时显示一些有用信息，或对用户输入的数据进行一些基本的有效性检查。当用户单击一个链接时，我们甚至可用JS指示浏览器完成一个确定的动作。简而言之，JavaScript是实现页面内容与用户交互的手段，使用户能与页面进行充分的交互。

听起来有点空洞(且有点乏味)，不是吗？大概最好的方法是立即深入进去。

1. 收集需求

在开始编码前，要确信对要构建的系统有所了解。需求收集对一个客户项目是非常有益的(在第1章已介绍过)，最小的开发项目也一样，能从一定程度的需求分析中获益。对要构建的系统和要达到的目标有更好地了解之后，编码工作会更快和更有效率，这两个方面会使您的客户和您自己非常愉快。

因此下面将对我们所要做的工作进行简单梳理。

- 在XHTML文档的head部分有三个link元素，其中包含三个控制显示的CSS文件：一个永久样式表(main.css)和两个备选样式表(contrast.css和极华丽的hot.css)。
- 因此在该文档顶部是一个含三个锚的列表，每个锚对应一个不同的样式表。必须承认，这些锚的用处大概与沙漠中的道路图一样，但我们很快就会做一些改变。

那么我们的函数究竟完成什么样的功能？理想情况，当用户单击一个链接时：

(1) 该函数应该遍历XHTML中位于head内的每一个link元素，并检查那些指向样式表且有标题的link元素。

(2) 如果该link元素与用户选择的链接匹配，应把它设置为“激活的”CSS。

(3) 否则应把该link元素设置为“非激活的”，以阻止浏览器加载该样式表。

(4) 一旦函数完成激活link元素的设置，应该记住用户的选择。用户选择的样式表在用户浏览整个站点的过程中一直保持为激活的，并且在以后的浏览会话期间用户回到该站点时仍应记住该选择。

了解了这个函数的整个功能了吗？这个方案最终将涉及大量的技巧和一些令人陶醉的思想，但在这里不准备深入讨论。

2. 构建切换函数

在头脑中有了明确目标后，就可以开始构建样式表函数了。首先创建一个新文件scripts.js，并在XHTML的头部包含下列标记：

```
<script type="text/javascript" src="scripts.js"></script>
```

与用link元素包含外部CSS文件类似，在这里用script元素引用一个外部JavaScript文件。在这个JavaScript文件的第一行将开启CSS切换器。如果说JavaScript语法有点难懂，那么不必担心。我们将略微分析一些被突出显示的部分。

```
// activeCSS: Set the active stylesheet
function activeCSS(title) {
 var i, oneLink;
 for (i = 0; (oneLink = document.getElementsByTagName("link")[ i] ); i++) {
   if (oneLink.getAttribute("title") && findWord("stylesheet",
oneLink.getAttribute("rel"))) {
   oneLink.disabled = true;
   if (oneLink.getAttribute("title") == title) {
     oneLink.disabled = false;
   }
  }
 }
}

// findWord: Used to find a full word (needle) in a string (haystack)
function findWord(needle, haystack) {
return haystack.match(needle + "\\b");
}
```

在这一小段代码中有两个JavaScript函数：activeCSS( )和 findWord( )，它们主要完成一些具体的功能。每个函数包含一系列送给浏览器处理的指令。例如，在调用activeCSS( )时将完成下列功能。

(1) 给函数传递一个参数(或变量)，这个参数表示要“激活”的样式表标题。

(2) 该函数获取到一个包含文档中所有link元素的列表(document.getElements ByTagName("link"))，并对这个列表进行遍历，检测链接的标题是否与函数的参数匹配。

(3) 当找到一个匹配时，就对其rel属性进行检测看是否出现单词stylesheet。在这里用函数findWord( )对rel进行全字匹配搜索。这意味着，如果有人在一个link元素中偶然输入rel＝"stylesheets"或类似的错误，这些链接将被忽略。

(4) 满足第二步标准的每一链接都被设置为非激活的(oneLink.disabled = true;)。

不可否认，这是对JavaScript函数语法的一点点注释。JavaScript是一个健壮的并值得学习的语言，但在此不能介绍它的精妙之处而不得不回到CSS的主题。但前述的程序清单说明了一些在上述代码中起作用的高级概念，也为有兴趣进一步研究JavaScript优美语法的朋友提供了一个良好的起点。

虽然可以用这两个函数切换CSS，但在它们被标记调用之前仅仅是潜伏的。当用户从#switcher列表中选择一个链接时，希望能触发这个切换器。触发的最佳位置是放在样式切换器列表的锚内。

```
<div id="switcher">
 <ul>
  <li id="style-default"><a href="styleswitch.html"
onclick="activeCSS('default'); return false">Default style</a></li>
  <li id="style-contrast"><a href="styleswitch.html"
onclick="activeCSS('Higher Contrast'); return false">Higher Contrast
</a></li>
  <li id="style-hot"><a href="styleswitch.html"
onclick="activeCSS('Gratuitous CSS'); return false">Gratuitous CSS</a>
</li>
 </ul>
</div>
```

这里引入的onclick属性称为一个事件处理程序。当用户执行某个确定的动作或“事件”(在这里就是单击鼠标)时，属性值中包含的JavaScript就被触发。因此在前述的示例中，当onclick处理程序检测到用户单击这个锚，它就触发activeCSS( )函数。

说明：

使用这些事件处理程序(如onclick、onblur、onmouseover等)类似于依赖样式属性——这些内联属性破坏了结构与行为的分离并很容易增加维护和支持的成本。与其通过编辑XHTML来反映JavaScript中的任何变化，不如用更现代的JavaScript生成链接所需的事件处理程序，这样也保持了标记与脚本的必要分离。对此感兴趣的读者可参考Peter-Paul Koch’的*Separating behavior and structure* (http://digital-web.com/articles/separating_behavior_and_structure_2)。

仔细研究这三个不同的事件处理程序会发现，每个对activeCSS( )的引用有些细微的差别，括号内的样式表(该链接要激活的)标题不同。这就是前面提到的参数，这是一个文本串，activeCSS( )函数用它比较每个link元素的标题。

说明：

您可能注意到，在调用activeCSS( )函数之后，onclick处理程序包含一些额外的文本：return false;。这在切换器中起到非常小的作用(但为了完整性而被保留)。它告诉处理程序不要跟随在锚的href属性中引用的URL。否则在用户单击任意一个链接后将进入styleswitch.html页面。

我们仅仅研究在单击某个链接时会发生的情况。假定用户选择第三个锚，即包含activeCSS('Gratuitous CSS')引用的onclick处理程序。

(1) 三个link元素被编译到一个数组，函数循环处理每一个link元素。记住，只检查包含title属性且rel属性含stylesheet的link元素。因此删除了contrast.css和hot.css。

(2) 第一个link元素的标题为Higher Contrast。该函数把这个link元素置为非激活的，且一直保持为非激活状态，因为它的标题与函数的参数(Gratuitous CSS)不匹配。

(3) 第二个link元素的标题为Gratuitous CSS。该函数先把它置为非激活的，但由于它的标题与函数参数匹配，又被重新激活。

图5-7显示了每个备选样式表的结果。

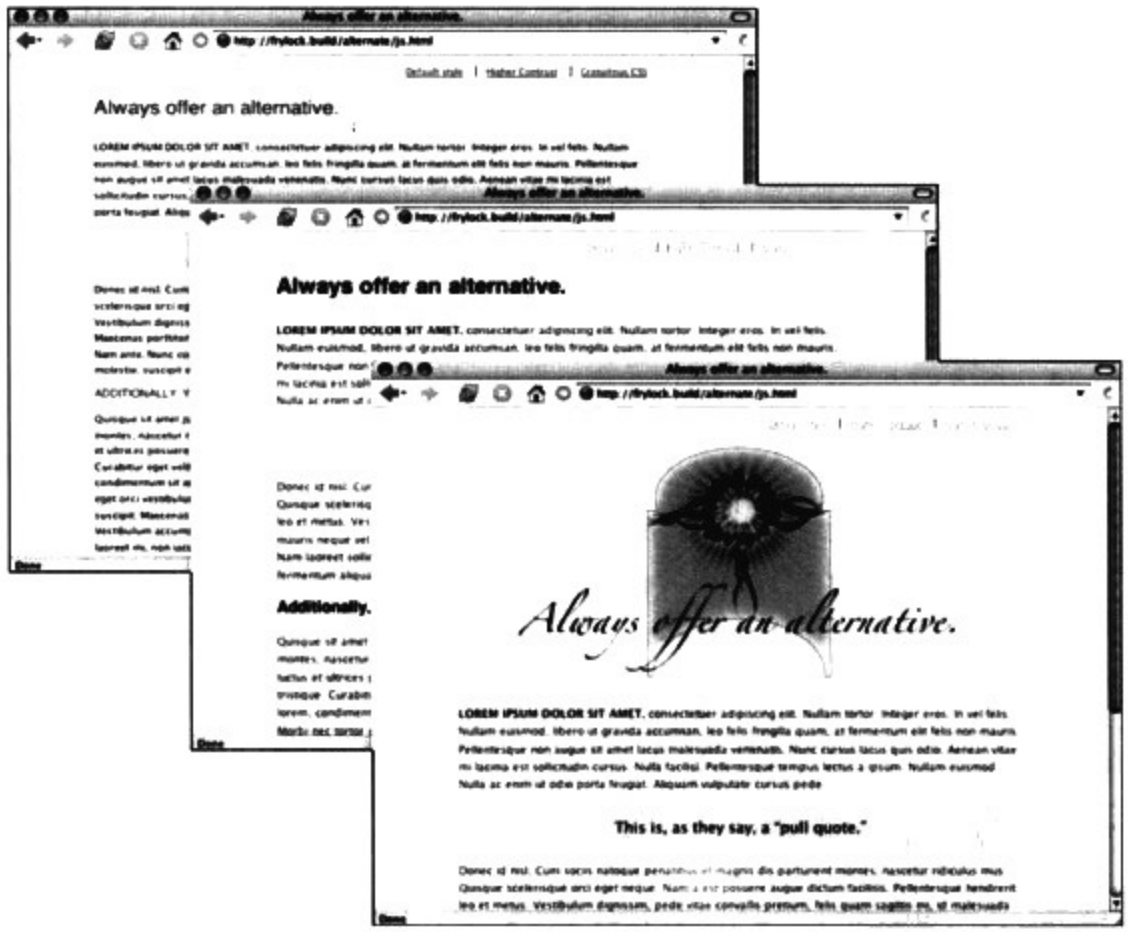


图5-7　通过JavaScript驱动的样式切换器，用户可以选择一个最满足需求的样式

虽然已成功创建了一个函数来切换不同的样式表，但这只完成了一半工作。如果用户刷新该页面或选择了新的备选样式表后离开了当前页面，选择的样式表将被忘记并恢复为默认的样式表。下面我们在JavaScript函数内申请一小片内存。

3. 设计一个JavaScript cookie

从这个没有完成的CSS切换器可以看到，一旦离开或刷新某个页面，浏览器似乎不能

记住有关该页的任何东西。这是由于设计造成的。HTTP标准(这是一个把Web页面从服务器传递到浏览器的协议)被设计为“无状态”的。这意味着，每次访问一个页面时Web服务器都认为是第一次访问。令人高兴的是有一个方法可解决这个问题。这种方法被称为cookie。

您可能知道，一个cookie是一个很小的文本文件，Web服务器把它送到用户的浏览器，它包含有关用户浏览期间的一些重要信息。cookie可能包含用户的偏好、注册信息、放在在线购物车中的物品，等等。一旦浏览器收到一个cookie，它会把这些信息保存在用户的计算机上；每当用户离开该站点，浏览器就会把该cookie发送回Web服务器。

我们可以用JavaScript设置和读取cookie，因此在已完成的函数中添加更多的JavaScript函数可以构建一个改进后的样式表切换器，这个切换器可以保存用户的偏好。

首先需要设置一个包含用户偏好的cookie，然后启动该cookie使之能被读取。因此在script.js文件中添加一个新函数setCookie( )：

```
// Set the cookie
function setCookie(name,value,days) {
  if (days) {
    var date = new Date();
    date.setTime(date.getTime()+(days*24*60*60*1000));
    var expires = ";expires="+date.toGMTString();
  } else {
    expires = "";
  }
  document.cookie = name+"="+value+expires+";";
}
```

现在，只需在原来的activeCSS( )函数添加一行(setCookie( )…)就可以把用户的偏好存储在用户计算机上的cookie中：

```
// Set the active stylesheet
function activeCSS(title) {
  var i, oneLink;
  for (i = 0; (oneLink = document.getElementsByTagName("link")[ i]); i++) {
    if (oneLink.getAttribute("title") && findWord("stylesheet",
oneLink.getAttribute("rel"))) {
      oneLink.disabled = true;
      if (oneLink.getAttribute("title") == title) {
        oneLink.disabled = false;
    }
   }
  }
```

```
    setCookie("mystyle", title, 365);
}
```

有了这一行，工作就完成了一半。setCookie( )有三个参数：cookie的名称(以后可以引用)、要存入cookie的值和cookie过期的天数。前面的小段代码创建了一个名为mystyle的cookie，要设置的值是activeCSS( )函数的参数title。这意味着，如果用户选择了一个链接，这个链接在onclick处理程序中指定了activeCSS('Higher Contrast')(即以Higher Contrast为title参数调用activeCSS)，则cookie(mystyle)的值将为Higher Contrast。

说明：

在setcookie( )函数中，指定cookie过期天数是可选的。由于这个参数是可选的，因此我们可以完全不考虑这个参数。然而，省略这个参数将导致setCookie( )函数设置的cookie(mystyle)在用户会话期结束后过期，结果导致用户一关闭浏览器就会丢失他的偏好信息。在前面的示例中，cookie(mystyle)过期的天数被设置为365天(或一日历年)。

通过对setCookie( )的独立调用，就可以把用户从样式列表锚的列表中所做的选择成功地存储起来。但是如何读取这个cookie并获得偏好信息呢？简单地把下面的行添加到script.js文件中即可：

```
window.onload = initCSS;

// initCSS: If there's a "mystyle" cookie, set the active stylesheet
when the page loads
function initCSS() {
  var style = readCookie("mystyle");
  if (style) {
    activeCSS(style);
  }
}

// Read the cookie
function readCookie(name) {
  var needle = name + "=";
  var cookieArray = document.cookie.split(';');
  for(var i=0;i < cookieArray.length;i++) {
    var pair = cookieArray[ i] ;
    while (pair.charAt(0)==' ') {
     pair = pair.substring(1, pair.length);
  }
    if (pair.indexOf(needle) == 0) {
```

```
        return pair.substring(needle.length, pair.length);
      }
    }
    return null;
}
```

把这最后几行JavaScript放到适当位置，就完成了最后的工作。新函数initCSS()有两个简单的任务。第一个任务：检查用户机器上是否有mystyle cookie(var style =readCookie("mystyle");)。如果有(if(style))，则以用户的cookie值为参数调用activeCSS( )函数。

第二个任务隐藏在一个相当不起眼的行“window.onload = initCSS;”中，这一行的工作很繁重：当文档在用户浏览器中加载完毕后触发initCSS( )。现在用户在站点的页面之间切换时，或用户在以后的会话期间回来时，随着每个页面的出现可以立即得到cookie(mystyle)。当用户在样式切换器中进行选择后，页面会记住他们的选择，使用户不仅能裁减个别页面也能裁减整个站点以满足用户的浏览需求。

程序清单5-6列出了整个JavaScript文件。

程序清单5-6：启动由JavaScript驱动的CSS切换器的完整script.js文件

```
/*
   Onload
*/
window.onload = initCSS;

// initCSS: If there's a "mystyle" cookie, set the active stylesheet
when the page loads
function initCSS() {
   var style = readCookie("mystyle");
   if (style) {
     activeCSS(style);
   }
}

/*
 Switcher functions
*/
// activeCSS: Set the active stylesheet
function activeCSS(title) {
  var i, oneLink;
  for (i = 0; (oneLink = document.getElementsByTagName("link")[ i]); i++) {
    if (oneLink.getAttribute("title") && findWord("stylesheet",
oneLink.getAttribute("rel"))) {
```

```
        oneLink.disabled = true;
        if (oneLink.getAttribute("title") == title) {
          oneLink.disabled = false;
        }
      }
    }
    setCookie("mystyle", title, 365);
}

// findWord: Used to find a full word (needle) in a string (haystack)
function findWord(needle, haystack) {
    var init = needle + "\\b";
    return haystack.match(needle + "\\b");
}

/*
    Cookie functions
*/

// Set the cookie
function setCookie(name,value,days) {
    if (days) {
      var date = new Date();
      date.setTime(date.getTime()+(days*24*60*60*1000));
      var expires = ";expires="+date.toGMTString();
    } else {
      expires = "";
    }
    document.cookie = name+"="+value+expires+";";
}

// Read the cookie
function readCookie(name) {
    var needle = name + "=";
    var cookieArray = document.cookie.split(';');
    for(var i=0;i < cookieArray.length;i++) {
      var pair = cookieArray[ i] ;
      while (pair.charAt(0)==' ') {
       pair = pair.substring(1, pair.length);
      }
      if (pair.indexOf(needle) == 0) {
        return pair.substring(needle.length, pair.length);
      }
    }
    return null;
}
```

5.4.2 PHP方案

JavaScript解决方案有个相当大的缺陷。我们怎么知道在用户机器上有JavaScript？我们几乎听到了嘲笑声？什么样的浏览器没有这样超酷的语言？但实际上这样的浏览器是相当多的。

互联网统计数据表明，所有的Web用户中有6%~10%的用户浏览时不会使用JavaScript。不论浏览时没有使用JavaScript的人有多少，为什么要把他们排除在外而不能访问您的站点呢？特别是可以利用服务器端的编程技术非常容易地复制相同功能模块来解决这个问题，我们马上就将介绍这种方法。

与其依赖对用户可能有用也可能没有用的客户端代码，不如创建一个驻留在Web服务器端的、处理样式切换的脚本。由于我们将在服务器端的环境工作，因此不必担心JavaScript是否在用户计算机上可用。只要用户能接收到cookie，服务器端的脚本就能很容易地处理样式表切换逻辑。

当然，服务器端的编程语言几乎与本书作者一样多。对这个项目我们使用PHP(http://www.php.net)。这是一种非常流行、源码开放的编程语言，在非常多的现代Web服务器上可用。由于PHP非常普及、速度快、功能强大，因此它是本章实验的最好选择。

说明：

当然，如果在Web服务器上没有安装PHP，就不能利用它。与服务器管理员联系以确定是否在您的服务器上安装了PHP，如果没安装，有丰富的资源指导如何安装PHP。

可能最佳的起点是PHP官方文件(http://www.php.net/docs.php)。老实说，对我们这些缺乏133t-ness分类知识的人来说，PHP安装指令(虽然写得相当清楚)有点令人望而生畏。如果发现自己在配置指令中迷失了，建议求助于您所喜好的搜索引擎。搜索“安装PHP Windows”或“安装PHP Mac”可得到大量(这是我们希望的)容易理解的结果。作为一种强大、健壮的编程语言，这也是PHP相当流行的另一证据。

另外，Mac OS X用户可以利用一些非常易于安装的包。我们推荐Server Logistics公司的PHP安装程序(http://www.serverlogistics.com/php4.php)，这是一个功能非常丰富的安装程序，但也有其他一些类似的安装程序。这些安装包的缺点是用户不能通过安装过程对PHP的配置进行控制。如果这种不需用户干预的安装不能吸引您，官方文件将成为您最好的参考文献。

1. 创建脚本

一旦安装了PHP并运行在服务器上，就可以开始下面的工作。首先修改XHTML，特别是包含不同样式表选项的无序列表。尽管前面已用onclick处理程序完成了这些繁重的工作，但在这里有些不同：

```
<ul>
  <li id="style-default"><a href="switch.php?style=">Default style</a></li>
  <li id="style-contrast"><a href="switch.php?style=contrast">Higher
Contrast</a></li>
  <li id="style-hot"><a href="switch.php?style=hot">Gratuitous CSS</a></li>
</ul>
```

现在所有链接都指向文件switch.php，但链接中的“?style=stuff”是什么意思？href中问号后的文本称为查询串，通过这个串可以给CSS切换器脚本传递参数。查询串参数一般以名/值对的形式出现，如下所示：

```
file.name?name=value
switch.php?style=contrast
```

要把多个名/值对传递给脚本，用“&”把它们链接起来，如下所示：

```
Switch.php?style=contrast&amp;font=serif&amp;css=cool
```

说明：

在HTML中，“&”有特殊的含义。它们表示字符实体(在HTML中代表特殊字符的代码)的开始。例如，“©”表示实体©，“™”在浏览器中显示为™，等等。当我们需要在查询串中出现“&”时，就需要用“&”，这是“&”实体的正确引用。否则这个HTML将失效，查询串的意思将改变。

很快我们就会看到这些参数在样式切换器中所起的作用，但现在我们需要创建switch.php文件，并把下列代码粘贴到该文件中：

```
<?php
$domain = "my-site-here.com";

if (stristr($_SERVER['HTTP_REFERER'], $domain)) {
  $bounce_url = $_SERVER['HTTP_REFERER'];
} else {
  $bounce_url = "http://$domain/";
}
```

```
setcookie('mystyle', $_GET[ 'style'], time() + 31536000);

header("Location: $bounce_url");
?>
```

实际上我们可以保证在接下来的50多个页面内的代码不超过80行。在JavaScript中需要编写一些定制函数，以处理样式切换器要解决的一些基本任务。但许多公共任务(如读、写cookie)已成为PHP的一部分。利用这些内置函数可以减少大量的臃肿代码并尽可能快地完成样式表切换器。

这段代码中值得注意的是倒数第三行：

```
setcookie('mystyle', $_GET[ 'style'], time() + 31536000);
```

在JavaScript中需要创建一个定制函数来设置cookie，但PHP为我们完成了这项繁重的工作。我们可以利用PHP的setcookie( )函数并在脚本中调用。与JavaScript的setcookie( )函数的相同之处也是要传递三个参数，最重要的是前两个参数。第一个参数和以前一样也是cookie的名称——在这里是mystyle。

第二个参数($_GET['style'])定义了要在mystyle cookie中实际存储的值。$_GET变量实际上是一个有名列表或联合数组，其中包含通过查询串传递给页面的所有参数。例如，假设用URL http://my-site-here.com/switch.php?style=hot&css=cool 调用页面switch.php。$_GET['style']的值为名/值对style=hot中等号后的内容，即hot，同理$_GET['css']返回的值为cool。结果setcookie( )函数创建了值为hot的cookie(mystyle)，这正是我们(和用户)所希望的。

可能需要对第三个参数(time( ) + 31536000)做一些解释，但也并不像表面上的那样令人望而生畏。函数time( )简单地返回当前时间，这是一个以1970年1月1日0时为起点的、以秒为单位的时间，也被称为“Unix历元”，这是处理基于时间任务的函数的共同参考点。一旦取到了当前时间，就添加一年的秒数(60秒×60分×24小时×365天＝31 536 000秒)。这样就得到了距设置时刻一年之后的时间，并把它作为cookie到期的时间。

一旦设置了cookie，就可把用户重定向回$bounce_url，这个变量是在文件开头设置的。在switch.php开头还有些额外的处理，以检查用户的调用网址(在调用switch.php前所访问页面的URL)。如果用户是从$domain上的页面 (if (stristr($_SERVER['HTTP_REFERER'], $domain)))调用switch.php的，则重定向到该页面。然而，如果其他站点决定直接链接到样式切换器，则把$bounce_url设置为我们的主页($bounce_url = $_SERVER['HTTP_REFERER'];)。

设置了cookie并把用户重定向到我们的站点后，接下来该做什么呢？我们需要设置一些逻辑来处理刚才创建的cookie。下面继续深入研究，看看有什么发现。

2. 处理cookie

这一步需要在XHTML中直接插入一些PHP代码——虽然不是很困难，但需要把标记文档转换为PHP服务器能理解的文档。为此，先把文件改名，然后把扩展名.html改为.php——如果系统管理员正确完成了这一步，就可把这个XHTML文档当作PHP文档了。

一旦更改了文件的扩展名，就可以把下列代码插入文档的head部分：

```
    link rel="stylesheet" href="main.css" type="text/css" />
    <?php
    if ($_Cookie[ 'mystyle' ]) {
    ?>
    <link rel="stylesheet" href="<?= $_Cookie[ 'mystyle' ]; ?>.css"
type="text/css" media="screen" />
    <?php
    }
    ?>

    <link rel="alternate stylesheet" title="Higher Contrast" href="contrast.
css" type="text/css" />
    <link rel="alternate stylesheet" title="Gratuitous CSS" href="hot.css"
type="text/css" />
```

当浏览器中加载这个标记文档时，这段PHP代码被插入head中。如果没有设置“mystyle” cookie(或值只是一个空串)，if{…}内的代码就不会执行。但如果出现了“mystyle” cookie，则一个新的link元素就会插入到标记中。下面继续深入展开。

根据无序列表#switcher内的查询串，“mystyle” cookie的两个可能值为hot和contrast。因此当单击href为switch.php?style=hot的链接时，link元素为：

```
<link rel="stylesheet" href="hot.css" type="text/css" />
```

这样就成功实现了PHP样式表切换器。构建了JavaScript切换器所描述的目标和概念之后，现在实现的解决方案就可以使用户自由地选择一个设计，不存在过多技术障碍。

5.5 超越浏览器的CSS

用现代桌面浏览器浏览您的页面时，您的页面看起来相当吸引人。我们已经有不同的方法允许用户改变站点的表现层。这个文档在浏览器之外的环境将会如何？当试图打印一

个超级样式设计时又会如何？

图5-8是Gratuitous CSS外观预览。

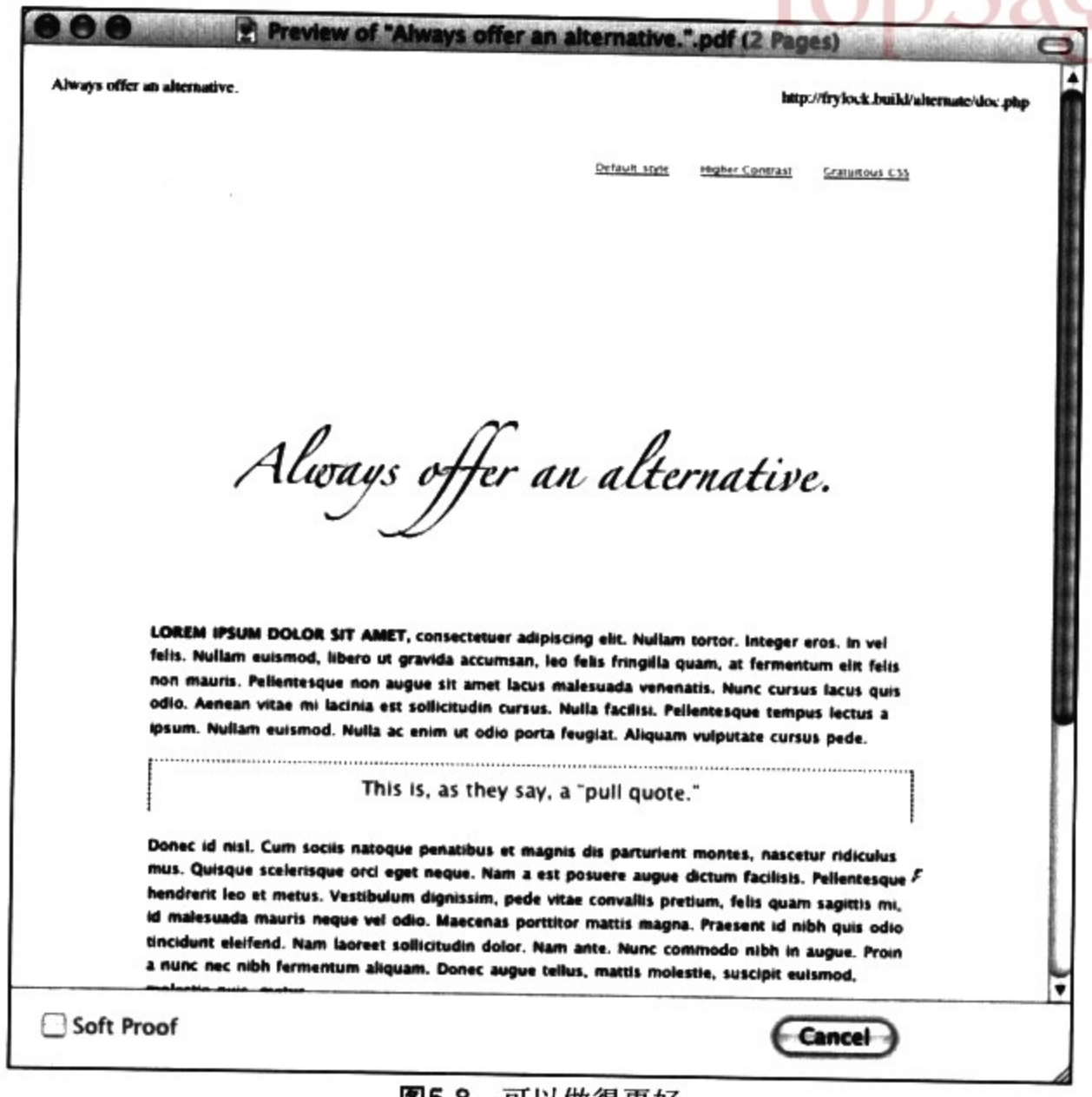


图5-8 可以做得更好

可以看到，打印时显示出的设计元素已不少了，但还远远不够。页面的背景色和图形没有被显示出来，因为打印机不会默认呈现背景色和图形。另外，管理页面布局的规则还完全没起作用。内容被压到了页面的下面，没有任何原因。顶部的空白区域没有任何目的，而且还剪裁掉了大量在页面上可见的文本。此外用于body拷贝的sans-serif小字体在屏幕上看起来很好(用户可以根据需要增加字体大小)，但在纸张上就没有这么理想。Serif字体通常在脱机浏览页面时用于增强易读性，比目前字体高一些并没有什么坏处。简而言之，这个打印版本不太吸引人。

我们应该从根本上为页面的打印版本进行单独的设计。这个设计不但强调样式的易读性，在美学上也不能有什么缺陷。但这带来不小的额外工作。页面具有仅用于打印的独立版本的时代仍没来临。要设计单一的页面和设计全面的页面保持同步，不仅费时还费资源，这常常需要编写很复杂的脚本、花大量时间进行手工编辑。

5.5.1 媒体类型：康复的开始

CSS规范的作者预见到了这个问题。他们引入了“媒体类型”这样一个概念，这是一种样式表分类手段，通过媒体类型可以把不同设计发送到不同的设备，如打印机、计算机显示器、屏幕阅读器、Internet掌上设备等。只需把三个link元素打上如“屏幕(screen)”这样的标记，就可以使这个设计只送到全图形的浏览器(如IE和Firefox)，这样就能避免前面遇到的一些不愉快情况。要达到这个目的，只需把link元素的media属性设置为screen：

```
    <link rel="stylesheet" href="main.css" type="text/css" media="screen" />
    <link rel="alternate stylesheet" title="Higher Contrast" href="contrast.
css"
    type="text/css" media="screen" />
    <link rel="alternate stylesheet" title="Gratuitous CSS" href="hot.css"
    type="text/css" media="screen" />
```

现在在打印视图下预览该文档，结果就大不一样了，如图5-9所示。

看起来并不理想，但这也是一个进步。通过在链接中添加属性media="screen"，就可把设计与使用环境(浏览器)结合起来。因此用户在不同媒体类型(如打印机)中浏览时，看到的就是没有任何样式的原始内容。

说明：

也可以给某个link元素指定多种媒体类型。例如，当以全屏浏览模式进行浏览时，Opera浏览器(http://www.opera.com)就会切换到“投影”媒体类型。因此忽略专门为“屏幕”媒体设计的CSS。如果希望在投影环境下重用屏幕样式表，可以将其添加在media属性后并用逗号隔开：<link rel="stylesheet" href="main.css" type="text/css" media="screen,projection" />。

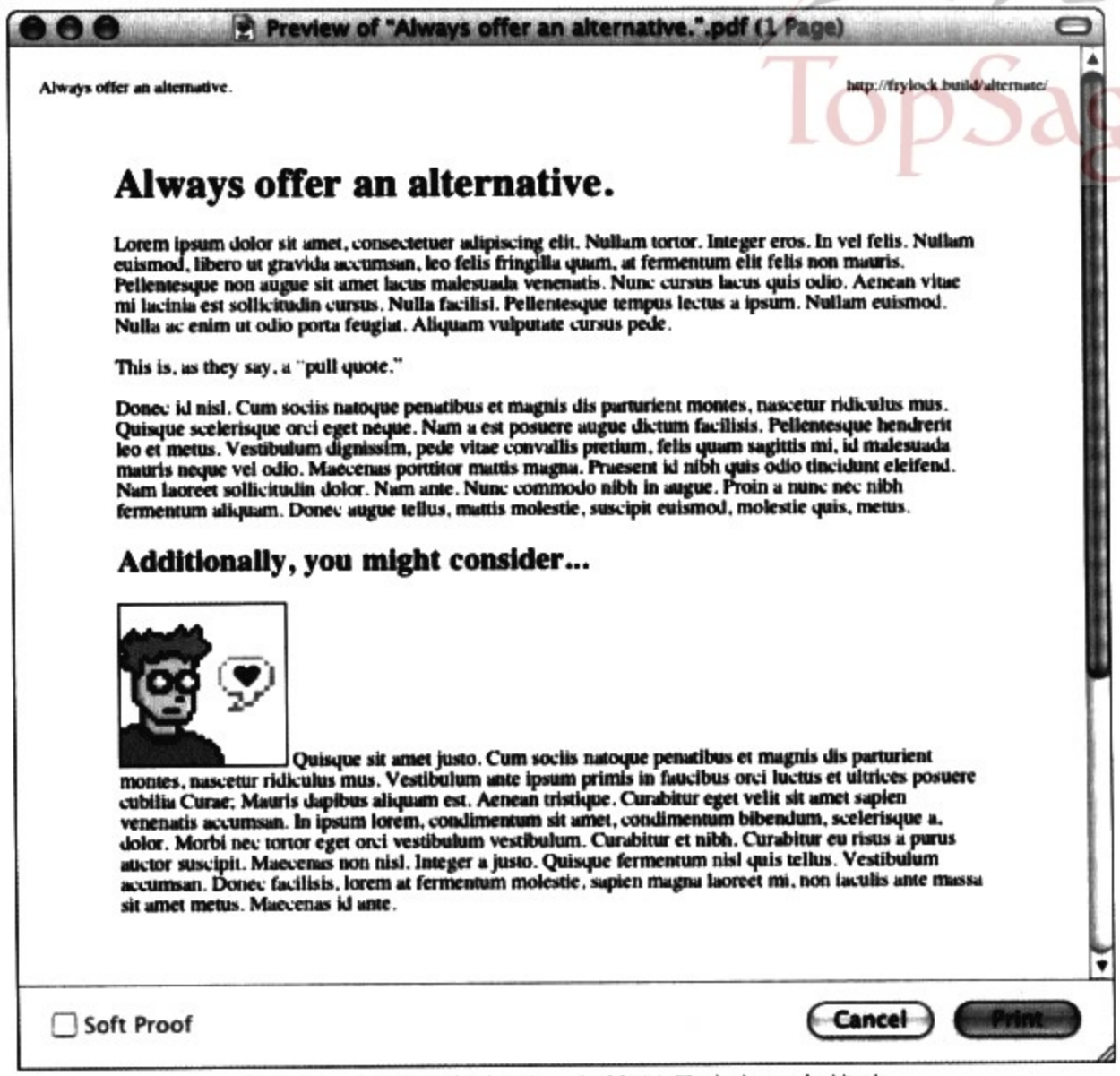


图5-9 打印文档很乱，但这只是建立一个基础

当然，可以指定样式表不能用于非浏览器设备，也可以指定样式表只能用于非浏览器设备。总之，如果针对屏幕的设计是健壮的，则没有理由出现无样式的打印结果。为此我们创建一个样式表print.css，如程序清单5-7所示。

程序清单5-7：print.css样式表

```
body {
  background: #FFF;
  color: #000;
  font: 12pt/1.4em Georgia, Garamond, "Times New Roman", Times, serif;
}

h1, h2 {
  font-weight: normal;
  margin: 1em 0;
  padding: 0;
```

```
    text-transform: small-caps;
}

img.portrait, #switcher {
    display: none;
}

#blurb {
    background: #CCC;
    border: 1px solid #999;
    float: right;
    font: 16pt/1.5em Helvetica, Arial, Geneva, Verdana, sans-serif;
    margin: 0 0 1em 1em;
    padding: 1em;
    text-align: right;
    text-transform: small-caps;
    width: 10em;
}
```

相当简单，对吗？在创建用于打印的样式表时，可以使用全书所介绍的相同语法和策略。不论是应用到浏览器还是纸张上，它仍然是CSS。不过得承认，在为打印设计时需要考虑下列情况：

- 字体大小控制——用点控制字体大小，这或许是针对打印的样式表最吸引人的地方。虽然点是字体大小的绝对度量，但可以用点显示打印样式。在针对屏幕设计时，要避免使用点带来的麻烦，因为浏览器对点大小的呈现是不一致的。然而针对打印使用点是比较理想的。
- 剔除无用部分——标记的某些部分可能不必在打印结果中体现。例如，您可能希望在打印时不打印那个丑陋的杯子。当然针对浏览器的样式表切换器，所有链接完全不用点。针对打印的样式表，可以简单地指定img.portrait, #switcher { display: none; }。通过媒体类型的帮助，这两个元素在屏幕上仍然可用，但在针对打印的版本中则被删除。

创建了针对打印的样式表后，把它包含在文档的head部分。和往常一样，可以用一个链接来实现，但要额外注意的是，要指定正确的媒体类型——即print。

```
    <link rel="stylesheet" href="main.css" type="text/css" media="screen" />
    <link rel="stylesheet" href="print.css" type="text/css" media="print" />

    <link rel="alternate stylesheet" title="Higher Contrast" href="contrast.
css"
```

```
type="text/css" media="screen" />
<link rel="alternate stylesheet" title="Gratuitous CSS" href="hot.css"
type="text/css" media="print" />
```


当我们再一次打印时，结果看起来就更好一点。这次，忽略了针对屏幕的样式，而采用print.css来控制外观。从图5-10可以看出，这种假设相当准确。

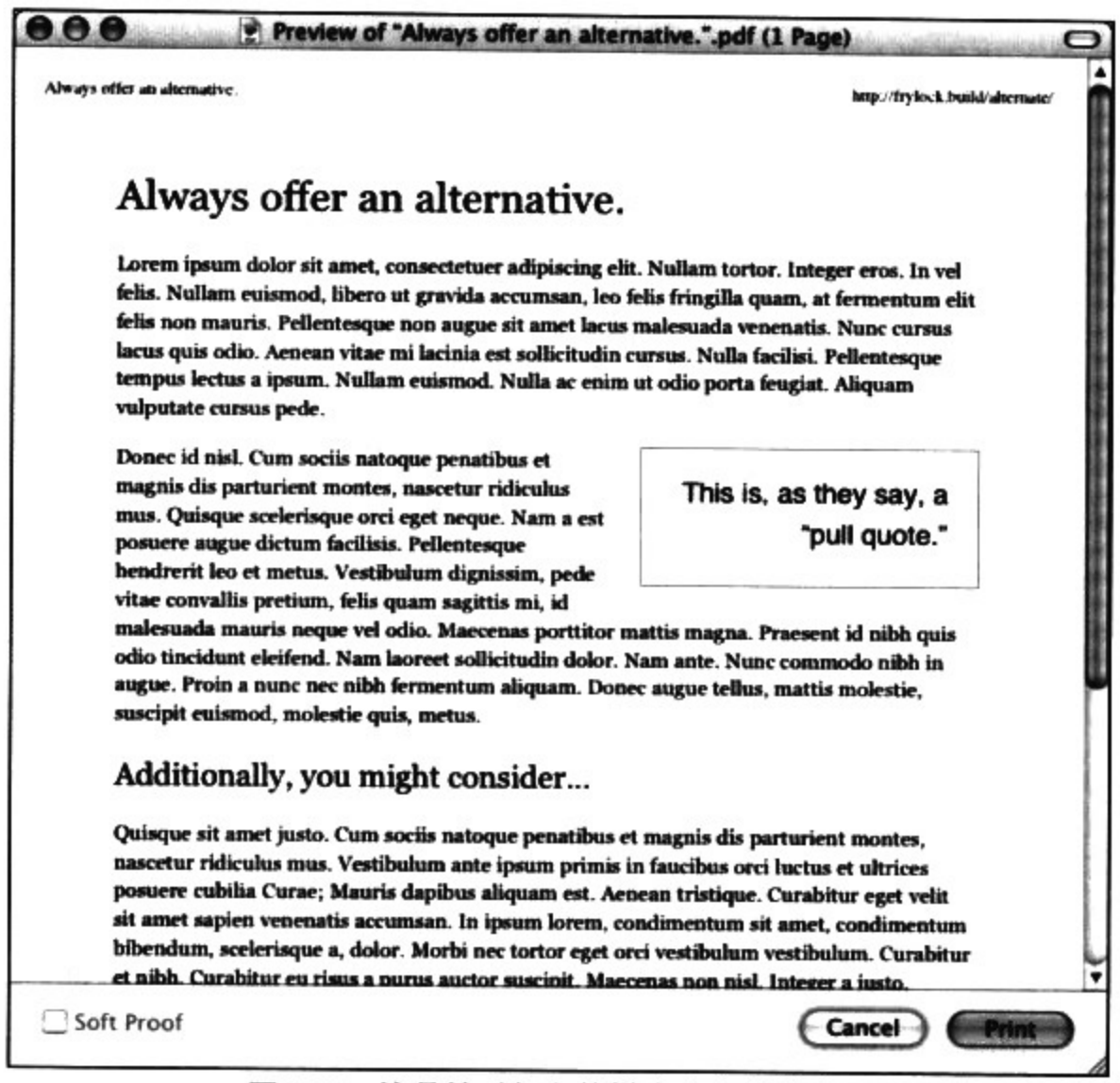


图5-10 这是针对打印的样式表的控制结果

极小的sans-serif字体已被更大的serif字体取代。当然，您可能不想将其设置为Model-T字体，因此可以为页面的打印版选择更吸引人的Garamond或Georgia字体。虽然在前面设置#blurb段落时，把一行设置为全部是一列的块，在这里我们可以用浮动模型把它从文档流中拉出来，并且使上引号的效果更好。

在针对屏幕进行设计时，所有这些情况都会独立地出现。本质上，使用媒体类型可以创建两个独立的、截然不同的页面视图：屏幕版(表现力丰富)和脱机打印版(注重内容)。一个标记文档被显示在多个设备上。样式表实现了对设备独立的承诺。

5.5.2 选择问题

既然已经实现了针对媒体的设计，某种程度上又回到了前面遇到的选择问题。我们使用户可以选择更适合他们的、针对屏幕的设计，但我们把一个针对打印的样式表强加给他们却使他们不能选择这样的样式表。难道用户必须牺牲这种非屏幕媒体的选择？

回答很简单："不必！"。我们可以回到JavaScript驱动的和PHP驱动的样式切换器，并添加一些针对打印的样式。当然，如果媒体类型很多，脚本(和呈现给用户的界面)将相当大且维护相当困难。而我们需要的是优美的、可扩展的方案，使我们针对多种媒体类型能非常容易和快速地管理备选样式，且不牺牲易用性。

5.6 Stuff and Nonsense：创建一个更好的切换器

幸运的是一些具有创新精神的人们已经提前想到了这些问题。进入Stuff and Nonsense网站(http://www.stuffandnonsense.co.uk)，这是一个位于英国威尔士的设计工作室，其主页如图5-11所示。快速浏览整个工作室的项目(http://www.stuffandnonsense.co.uk/work)后，会产生两个不同的认识：第一，工作室完成了诸如Disney和World Wildlife Fund(世界自然基金会，WWF)这些全球知名品牌的网站设计，设计方案非常漂亮和耀眼；第二，站点设计是由层叠样式表驱动的，是以有效XHTML为基础构建的。

显然，Stuff and Nonsense完全采用了Web标准。但浏览了整个网站后，明显体会到它对用户需求的尊重。Accessibility页面最有特点的地方是邀请用户进入页面Customise the look of this site (http://www.stuffandnonsense.co.uk/company/iotbs)的链接，如图5-12所示。在这个页面，用户不仅可以针对浏览器也可以针对打印选择不同的样式选项。不论是喜欢以小字体的sans-serif还是以更大一些的serif阅读打印的文档，用户都可以选择。此外，用户的偏好被存储在cookie中，以致这些偏好可以保持到整个站点访问结束。用户可以浏览或打印站点的任何一个页面，并在整个访问期间都以他最满意的方式显示。

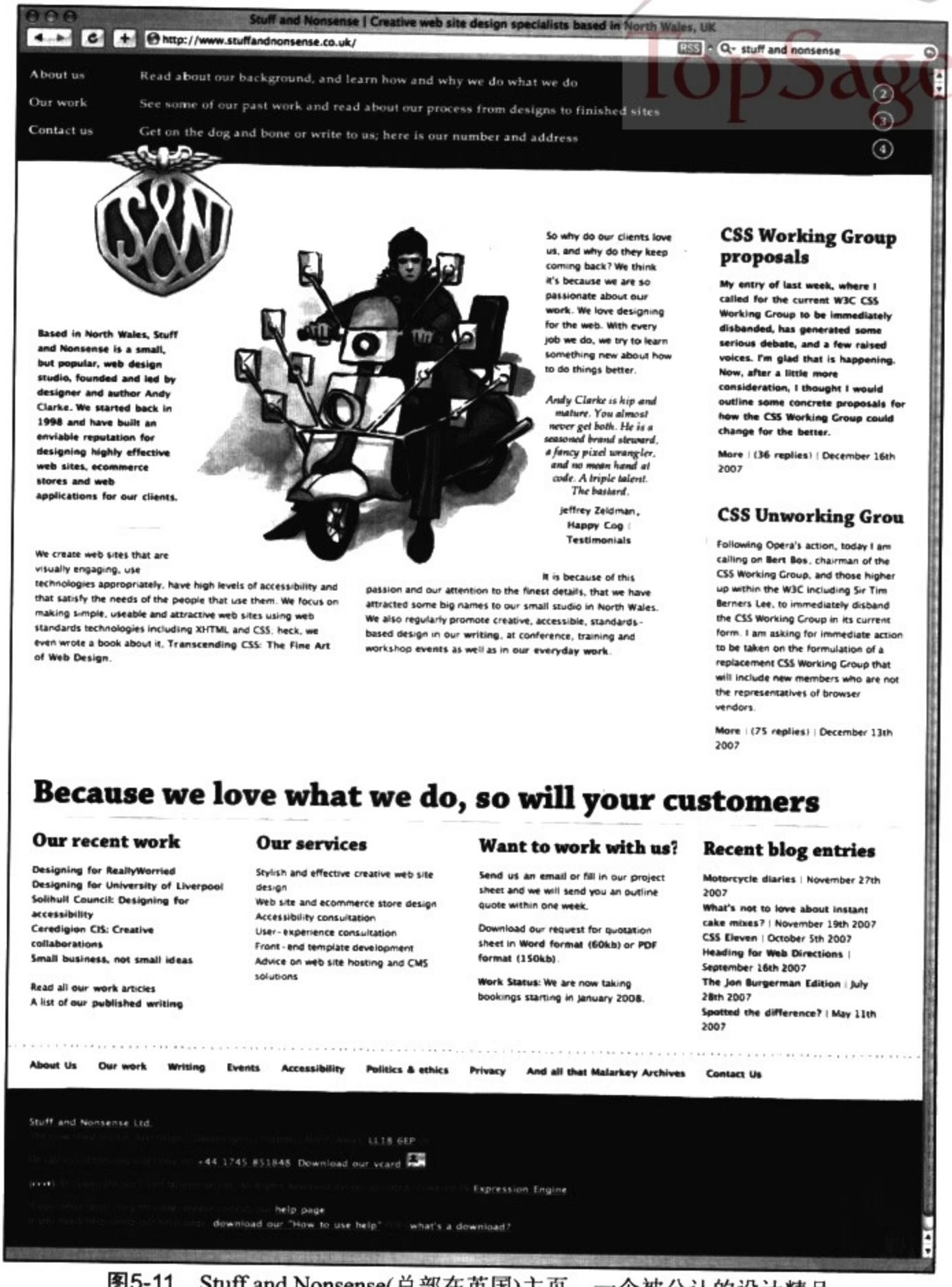


图5-11 Stuff and Nonsense(总部在英国)主页，一个被公认的设计精品

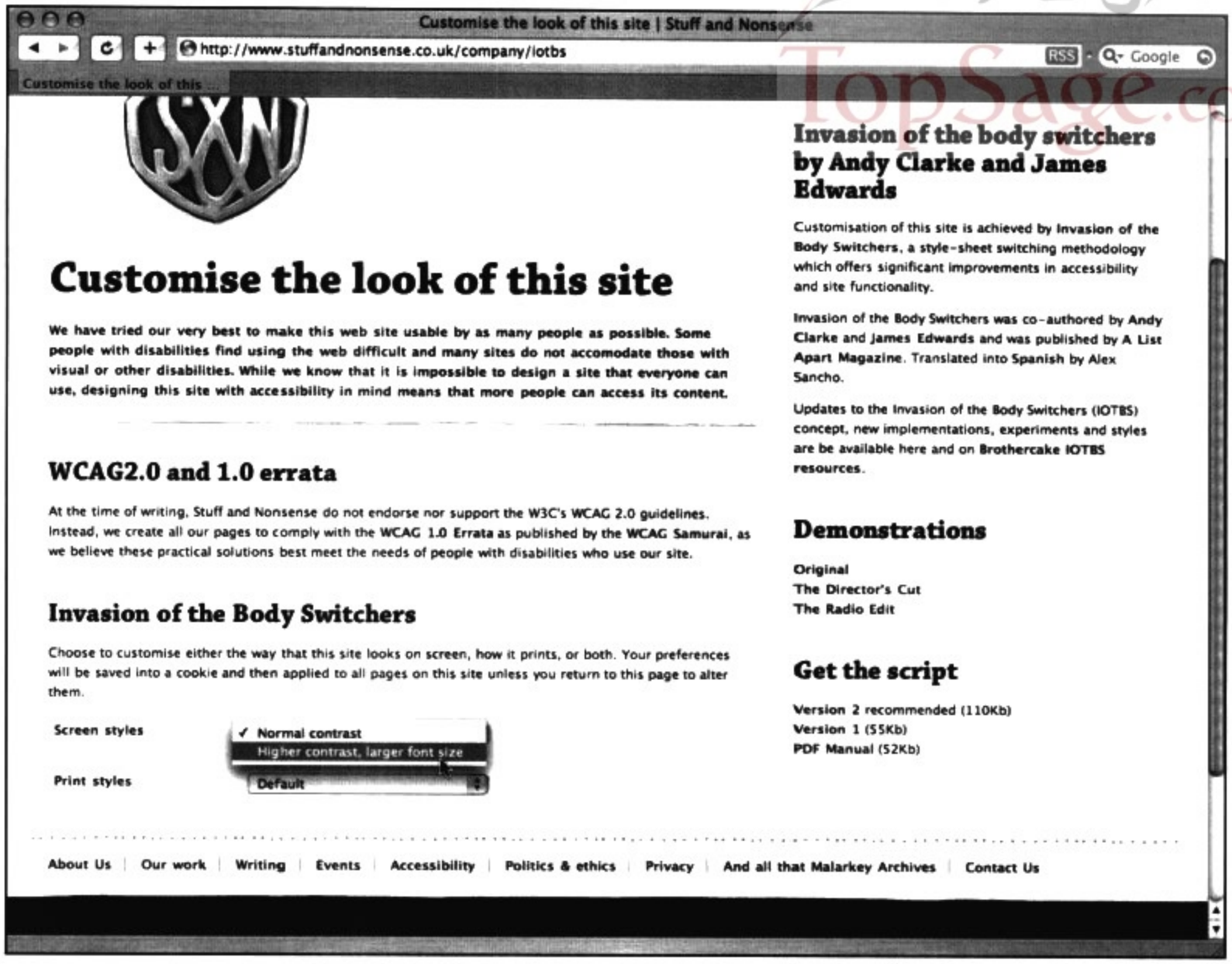


图5-12　样式切换器Invasion of the Body Switchers，通过一个简单的界面允许用户针对屏幕、打印或任何其他媒体类型选择不同的样式选项

这是一个命名奇特、功能丰富的样式切换器——Invasion of the Body Switchers (IOTBS)。其名称源于其切换功能的驱动方式——由body元素的class属性切换。它可能是相当完美的样式切换器，IOTBS的作者允许大家从http://stuffandnonsense.co.uk/resources/iotbs.html免费下载该切换器。由于非常容易安装和配置，IOTBS给站点所有者和用户提供最大的方便。它甚至能通过JavaScript生成整个界面，确保不给那些不能利用CSS切换的用户呈现非功能标记。

有了像IOTBS这样的工具，就可以充分利用针对媒体的样式表，充分发挥其威力。用户将对您提供的很容易安装的界面表示感谢。现在他们可以对不满足需求的粗边进行打磨，并把站点裁减成真正能使用的网站。

5.7 设计师Andy Clarke访谈

简单介绍IOTBS之后，现在介绍其中一个设计师Andy Clarke，他是Stuff and Nonsense创意总监。Andy是Web专业领域内不可多得的人才。Andy是多才多艺的设计大师、开发人员和作家，自1998年Stuff and Nonsense立项以来，他一直担任该项目的创意总监。创意设计人员占开发团队的一半，他们在Web站点引入如此独特的样式切换方案。Andy欣然接受我们的采访，并就可访问性、高品质设计以及两者的关系回答了我们的问题。

Q：Andy，非常高兴能和您进行一次简短的访谈。您是怎么步入Web设计这一行的？

A：哦，说来话长，我简单谈谈吧。大学毕业后(我学的是美术并获得了学位)，我从事过不同的工作，但都与艺术和工艺有关。我是英国第一批从事专业数字相机设计的人，在之前从事数字照片润饰，要知道这时还没有PhotoShop！我的艺术背景使我在工作中总是保持着富有创意的眼光，当我有机会成为伦敦的一名广告创意设计代理时，我抓住了这个机会。

Q：在Stuff and Nonsense工作室的客户列表中赫然列出了一些世界知名的公司名称，包括(但不限于)Disney Store UK 和 World Wildlife Federation UK。给人印象更深刻的是所有项目设计清晰，而且都是用XHTML/CSS实现的。为什么要采用Web标准？

A：为什么不呢？我没发现Web标准或访问性有什么问题。我相信它们是使工作能“正确”完成的必要部分。与Stuff and Nonsense客户合作中我学到的一件事，就是他们很少关心工作是“如何”完成的。他们所关心的是如何成功地满足目标用户的需求。

您提到Disney Store UK，公平地说，与大多数客户一样，他们并没要求实现一个与标准兼容的站点。但他们希望减少下载时间，同时使购物过程更快。采用与Web标准兼容的技术实现的设计满足了要求，并完美地达到了他们的目标。

Disney Store UK网站是用Karova Store平台(http://www.karova.com)开发的，这个平台不仅使表现层与站点的其余部分分离，而且采用的XML体系结构而不是以数据库为后台。通过XSLT把XML转换为XHTML，最终结果就得到了一个站点。该站点非常灵活，允许Disney Store UK在将来发布新的内容，包括通过RSS的反馈。现在大多数用户关心的不是“工具”，而是提供给他们的解决方案。Web标准提供了更多的解决方案，这就是为什么Stuff and Nonsense只按Web标准开发。

Q：你们所设计的东西那么智能那么小巧。我们认为那就是一个样式表切换器。能给我们介绍一下吗？

A：您是指Invasion of the Body Switchers (IOTBS)吧，就是我发表在List Apart杂志上的样式表切换器(http://www.alistapart.com/articles/bodyswitchers)？我不能把别人的荣誉据为己有。IOTBS背后真正的技术天才是我的好朋友James Edwards(http://www.brothercake.com)。他也是项目组成员，他根据我的创意实现了IOTBS。

Web标准的一个重要方面是对设计人员能力的要求，要求设计人员不更改Web页面的相应标记(HTML或XHTML)，而通过CSS实现页面的外在表现。这方面没有比Dave Shea's CSS Zen Garden(http://www.csszengarden.com)做得更好的了，通过使用CSS样式表才可能使该网站的同一页面有不同的设计。基于各种原因，样式表切换是必需的。或许客户希望给访问者提供可在固定宽度或充满整个窗口的“流动”布局之间进行切换的能力——设计师和站点构建者Dan Cederholm在站点SimpleBits(http://www.simplebits.com)中就提供了这样的选择。也可以采取给低版本浏览器的用户提供一个“可访问”的设计。这完全是可能的，有时目标和结果是一致的，有时只是一些讨厌的噱头。

服务器端和JavaScript样式表切换器已经出现好多年了。但Invasion of the Body Switchers与众不同的，它具有在屏幕、打印机和其他媒体样式之间独立切换的能力。所需一切仅是一个CSS和JavaScript文件。我为IOTBS感到非常自豪，我希望它能给更多的设计人员带来方便，这些与标准打交道的设计人员可以扩展他们的创意选择。

Q：我们发现它被广泛应用在Stuff and Nonsense 站点(http://www.malarkey.co.uk)，你们在其他专业的工程中使用过它吗？这对客户很重要吗？

A：对客户越来越重要的是给他们提供选择。像Invasion of the Body Switchers这样的样式表切换器可以用于给不同的用户群提供不同的设计主题。但用CSS的“display属性”也可以隐藏和显示内容。

这在最近几个针对年轻人的项目中发挥了巨大作用。利用CSS和IOTBS，我们不再需要对一个XHTML文档甚至整个Web网站编写三个或更多的版本。这缩短了开发时间，使我们的工作更有效率，最终为用户省了钱。每个人都很愉快。

Q：一些设计人员对用户能从根本上修改站点的设计感到不安。您对他们有什么要说

的吗？为什么让我们的用户对页面的设计有更大的控制权？

A：作为Web站点的设计人员或开发人员，我们需要记住是在为谁工作。当然是我们的客户，他们是我们的衣食父母，我们的最终工作是由站点的用户来评判的。他们越满意，我们的客户才能更满意，才更有可能给我们提供更多的机会。

Web与任何其他媒体不一样。对于电视，无论屏幕尺寸是多大，图形都会保持得很好。无论是CRT、LCD还是Plasma，是17"便携式还是52"宽屏，图形的显示效果都是一样的。对于Web而言，情况截然不同。对如何传递内容我们要有更多的控制，Web设计人员必须记住：用户的观点远比设计人员自己的观点重要。

Q：在网上闲逛了一会后发现，似乎在线发布了很多样式切换器。其中一些依赖客户端的JavaScript(如您们的一样)，而其他一些依赖一些后台的编码。两种方案各有什么好处？

A：现在您问到技术细节了。我只是一个粗浅的设计人员。有很多解决方案可以实现样式表切换器。一些是“服务器端”的(依赖后台语言，如PHP)；而另一些如Invasion of the Body Switchers则是“客户端”的，采用的是如JavaScript这样的语言。开发人员选择哪种解决方案取决于站点的运行环境和客户的特定需求。

这只是个人偏好，但由于样式表切换是一个“客户功能”，因此我偏向于采用客户端的解决方案。也就是说，我可以给你一个独家新闻，很快就会有Invasion of the Body Switchers的服务器端版本。

因此我冒昧地问一个问题：您的客户端切换器在众多切换器中脱颖而出的原因是什么？

Invasion of the Body Switchers采用新的样式表切换方案。我们的方案不抛弃常规“样式表”和“备选样式表”语法，但不会给我带来麻烦，因为：

(1) 许多浏览器没有实现本地样式表切换。

(2) 它们没有赋予可选择的备选样式表任何持续性。

其他解决方案使用<link/>元素和“样式表/备选样式表”语法，要依赖多个样式表。这带来额外的服务器调用，但更重要的是不能彼此独立地选择不同媒体样式。

Invasion of the Body Switchers使我们可以独立选择不同媒体类型并给站点用户提供一个简单界面。通过这个界面，用户可以选择他们喜爱的样式，所有选择都被保存到cookie直至他们改变了选择。

IOTBS是通过在页面的<body>标签内添加一个或多个唯一的类名来实现的。然后用后代选择符定义样式。最终结果使用户对Web页面的输出有更多的控制。

Q：非常有趣，您提到的“媒体类型”指的是什么？为什么了解CSS的Web设计人员需要关心媒体类型？

A：设计人员不仅要知道他们的工作从每个人的视角看是不完美的，而且还要知道人们会通过不同媒体访问Web内容。对从其他媒体转到Web的设计人员来说，要知道这一点有时是很困难的。有时这个媒体是我们的好朋友——计算机显示器；有时是机场的一个Internet小亭；有时是手提电脑、投影仪、甚至是移动电话。有些人发现在屏幕上阅读很困难而愿意打印出来再看。

在过去，要考虑所有这些不同的媒体类型，在成本上是不允许的，如果实在有这样的需求，则需要对不同的浏览设备设计不同的版本。但利用支持通用标准的技术，我们就可以创建只编写一次的内容，然后针对不同的媒体输出进行不同的样式控制，所有这一切都是通过神奇的CSS实现的。

Q：话说回来，我们希望了解有关设计过程更详细一些的情况。您通常是如何工作的呢？

A：首先我们要了解客户试图给他的用户传递什么样的信息。我们还要体会到公司的“个性特点”并考虑公司的品牌价值(即使他们自己并没有这样做)，这样我们的设计基调才能与公司的个性特点和品牌价值吻合。有效的Web设计能使客户和他们的用户进行有效的交流。这就是在考虑技术或创意问题之前首先要把重点放在交流的内容和如何交流的原因。

我们首先在纸上开发原型设计，用Photoshop 或Macromedia Fireworks完成从草图到布局的设计。这些布局一开始就是一些简单的连线框，从连线框构建标记指南，这一步通常也在纸上完成。我们的开发人员把标记指南作为XHTML的结构。

在之前我为我们的网站开发设计了一套命名约定，为 和类指定名称，它们与内容相关而不是与表现相关(#branding而不是#header，等等)。我们严格遵循这些约定，以便整个团队能理解一个特定CSS规则与哪些内容相关。我们还制定了图像命名约定，这也加快了开发过程。

然后把图纸上的布局转变为设计团队所需的产品，很少出现最终的Web页面与图纸布局不十分匹配的情况。每个阶段还必须经客户批准，我们总是按内部约定框架工作，这样保证了开发是尽可能高效的。

Q：那创作灵感呢？面对客户的严格基线，您从哪里获得灵感？

A：我是一个对流行文化非常上瘾的人。我喜欢各种侦探小说，比如Mickey Spillane的MikeHammer。我更喜欢漫画和漫画艺术家的作品，如Frank Miller 的*Sin City's*和Paul Chadwick的*Concrete*，这些都给我提供创作的激情。

我分析这些例子，发现它们的共同点都是为了激发用户参与而设计的，因此我把它们引入到不同的地方。我再分析其中包含的信息……并回过来分析Web中的主题。

Q：与其他人相比，您更经常遇到一些CSS问题吗？您是如何解决这些问题的？

A：对我来说CSS问题已越来越少，当出现某个问题时，通常通过Googling很快就能找到解决方法。很多人远比我聪明，马上就浮现在我脑中的就有Brothercake、Dave Shea (http://mezzoblue.com)、Doug Bowman (http://stopdesign.com/)和John Gallant。他们找到了解决浏览器错误和行为的方法，有些是我永远也想像不出的。当然，我有时也能修复一个或另一个浏览器中的问题，此时我会高兴得大叫：“用表格这将非常非常容易！”。但这样的情况已越来越少。

现在几乎每个CSS问题都能被修复或存在变通的解决办法。当一个未预料的CSS问题出现时，重要的是采取合理的方案，因为有时一个元素和另一个元素组合时会引发问题。

有效性非常重要，就连打开一个CSS文件之前我总是首先要确保自己代码是有效的。如果我的代码和CSS都是有效的而问题仍然出现，我会依次删除页面中的元素，以发现是哪个元素出现了问题。

许多浏览器错误是众所周之的，现在有专门解决这些问题的网站，如John Gallant的网站Position Is Everything (PIE，http://www.positioniseverything.net)。如果在PIE或其他网站上找不到答案，我建议向Eric Meyer (http://www.meyerweb.com)的优秀CSS-D邮件列表 (http://css-discuss.org)中的专家寻求帮助。只要能详细地描述问题，您很可能会找到解决问题的钥匙。

Q：您认为一个“成功”网站设计追求的是什么？您有一些特别好的设计原则吗？

A：我认为我不是评判一个设计是否成功的最佳人选，但我能倾听客户和他们的用户的反馈意见。对我来说最重要的是，该项目能给付我报酬的客户带来很好的经济效益。只有这样他们才能再给我项目。

当我回头再来看我所完成的许多设计时，我最喜欢的是那些更清晰、更简单的设计。没有设计的“仙丹”能把一个内容很差的网站转变为一个成功的网站。因此我和客户一起工作，确保把内容放在首位。

内容之外的工作总比试图把一个硬塞在一起的内容变为预期的设计或布局容易。这就是为什么在考虑设计的外观和感觉之前，我会在站点规划和内容上花更多时间的原因。

Q：还有什么需要和我们分享的吗？

A：“当我不知道用什么颜色的时候，我选择黑色。黑色有一种影响力：我依靠黑色来简化构造。”实际上这不是我的话，而是Henri Matisse的。

5.8 小结

对与媒体相关的CSS和三种不同样式切换策略有非常清楚的了解之后，我们已发现了不少内幕。和本书一样，本章就像景点相当丰富的景区的导游图。因此感觉很像一面镜子。其中的每个主题都可以扩展为一章，建议大家对每一个切换策略进行进一步的研究。您将使更多用户可以使用您的网站。

下一章我们将把本书所介绍的技术和策略应用到一个实际的网站设计中。

第6章 CindyLi.com的风险投资：博客修改

CSS是一个强大的工具，本章我们将看到Web网站CindyLi.com的概念是如何形成的、是如何实现的以及是如何用CSS创建(和操纵)CindyLi.com的。在本章我们将学习如何用CSS标记对网站和/或博客进行设计/重新设计的。要浏览该网站，可在浏览器中输入http://www.cindyli.com，如图6-1所示。

6.1 博客

博客(blog，是Web log的简写)是一种在线杂志，其条目通常按时间降序显示。一个博客可以是整个Web站点，也可以是添加到Web站点的组件或元素。

博客被认为是个人的，一些博客甚至是在线日记。大多数博客包含许多媒体元素，如文本、图形或图像、超链接，也包括音频和视频剪辑。在许多博客中还可以找到播客(由iPod 网络广播演化而来)。

建立博客的唯一规则(如果有规则的话)是所有内容应该与博客主题相关。许多博客的主人给读者提供在页面留言的能力，以便能和其他人讨论博客的主题。

在本章的示例(CindyLi.com)中，我们将学习如何设计一个简单的博客，然后用CSS增强这个博客。要创建自己的博客(和进一步学习)，可以访问下列链接：

- Technorati——http://technorati.com
- Blogger——http://blogger.com
- WordPress——http://wordpress.org

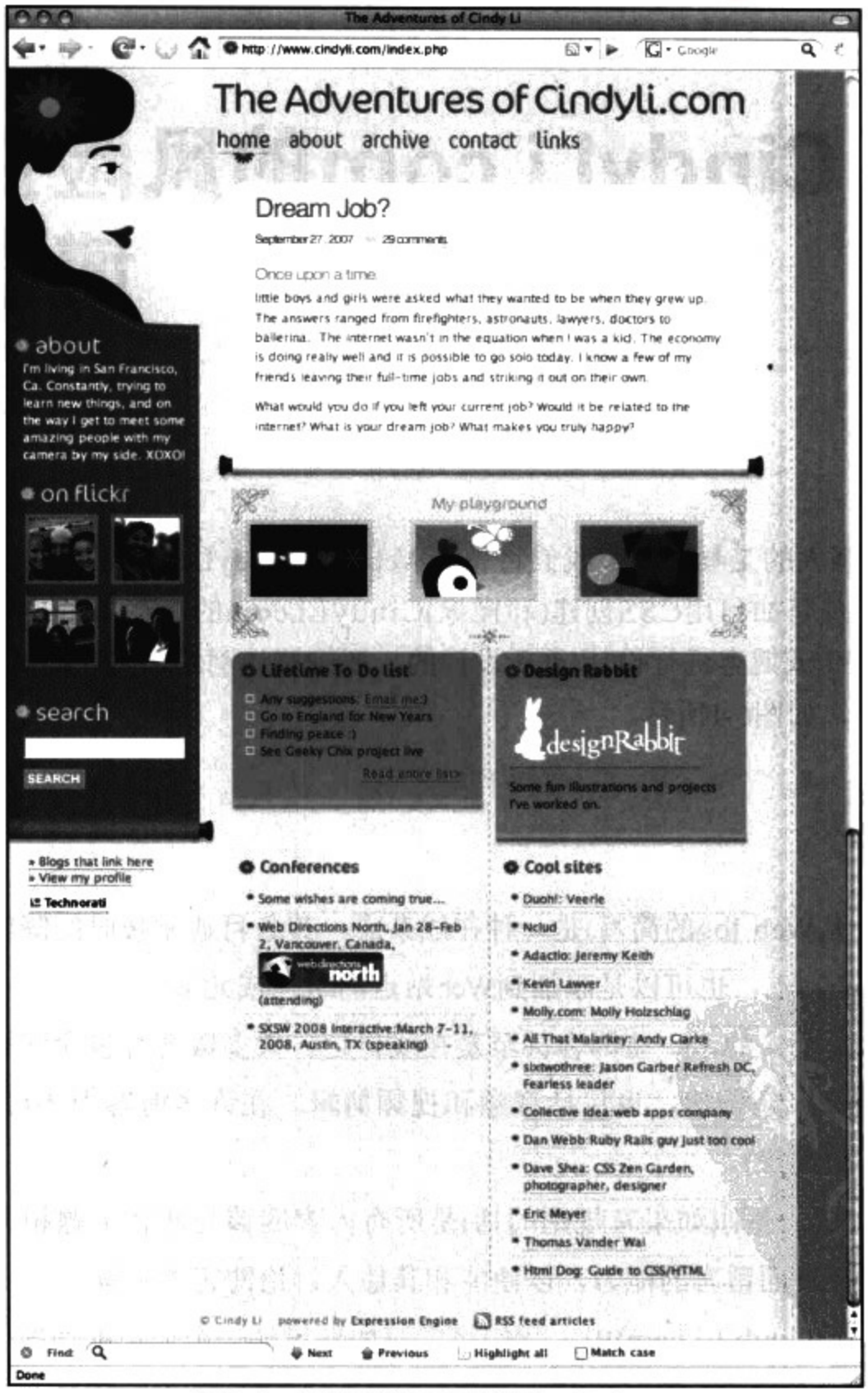


图6-1 已完成的Web网站CindyLi.com

6.2　CSS：Cindy Li 开博客

我们先简单回顾一下CindyLi.com的历史，以便我们了解该设计的演进过程。

CindyLi.com从纯文本博客发展为视觉丰富、功能强大的页面，包括博客主人(Cindy Li)的个人资料和一个供博客浏览者评论的lifetime to-do列表。通过使用CSS(和具有设计与作图的功能)，该网站具有极好的视觉概念，焕发了新的生命。

CindyLi.com在易于使用、趣味性和以真正艺术的方式使用博客功能等方面非常独到。它还捕捉到一些对博客主人很重要的信息，使它不仅功能强大而且交互性也很好。博客主人、艺术家Cindy Li在参加2006年SXSW(South By Southwest，德州在美国国土上的地理位置，指德州)的Interactive Fistival时深受鼓舞。Cindy为每次SXSW会议创建博客，每晚把它上传到一个以WordPress驱动的博客。她立即被突然大量增加的站点访问量所激发，非常高兴能交互地分享她的思想和观点。上述原因激发她把博客办得更大、更好，甚至利用它还提高了她在其他艺术方面的贡献。作为一个艺术家，重要的是博客不仅要体现她思想的精髓，还要以独特的方式在艺术上代表她，而且还要非常实用。

下面进入CSS的主题。计划是把一个lifetime to-do列表、一个博客名单、一个即将召开会议列表和图片组合成一个非常高端的look-and-feel，同时还要保持站点的趣味性。

我们在本章将看到，CindyLi.com开始不过是一个设计模板，我们可以自己下载这个模板，并在几分钟内可完成安装和配置。图6-2显示的是包含Cindy心血的博客设计模板。

6.3　设计要素

Web站点最重要(或至少是需要首先完成)的要素是站点布局和设计。没有周密的计划是无法创建一个网站的。一个好的计划既节省时间还能提供一些期望达到的目标。为了创建CindyLi.com，第一步就是要创建一个布局。布局可以与仅创建网站本身一样简单，可以是一个基于文本的博客，然后在此基础上发展。开始设计一个很小的布局，然后添加到站点，这并没有坏处。唯一应该采取的设计步骤是,一定要从一个博客工具着手，这有助于在学习中成长起来。

6.3.1　创建布局

一个好的设计(和设计的布局)通常是用Adobe的 Photoshop和Illustrator这样的工具开发的。虽然本书不涉及这些主题，但大多数CSS用户(特别是经验非常丰富的用户)对它们非常

熟悉。下列站点可供我们学习如何用Photoshop和Illustrator完成布局以及其他设计工作：

- http://tutorialoutpost.com
- http://photoshopzilla.com
- http://photoshopcamp.com

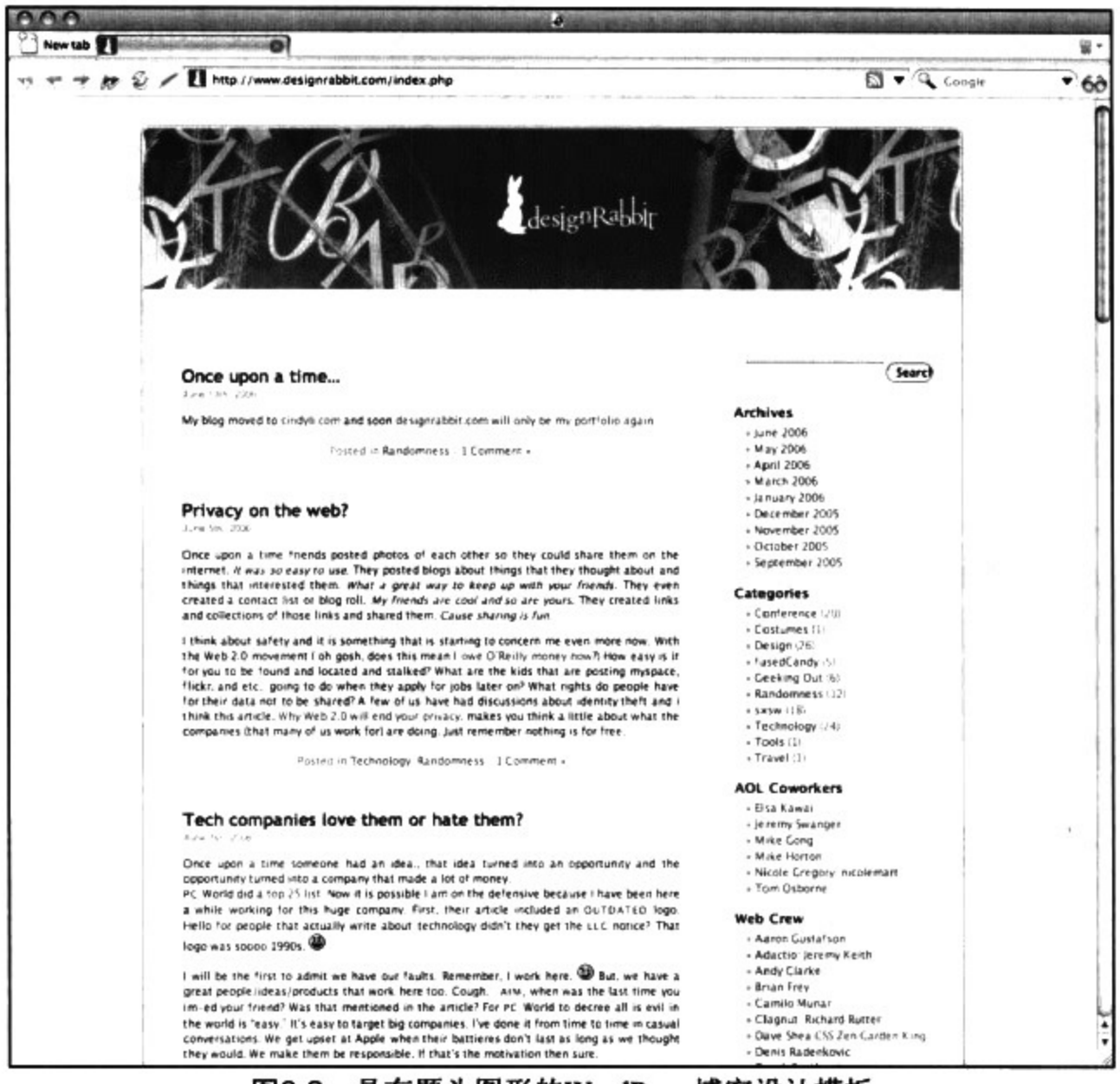


图6-2　具有题头图形的WordPress博客设计模板

以图形方式设计站点布局使我们能看到最终结果的视觉效果。这是一种很好的方法，这样在编码前可对布局进行修改(这意味着，在以后对设计不满意时，不必修改代码，只需修改图形即可)。

6.3.2　对设计进行布局

对于每个工程，甚至一个博客，设计人员必须列出网站应该具有的要素。这个详细列

表必须包含，在视觉和功能方面所需的内容。例如，CindyLi.com打算反映博客主人的美籍华人血统，因此要从整个布局上考虑色彩和元素。

总是要在项目早期确定设计或重新设计的需求。如颜色和徽标这样的参数在事先确定后，设计将变得更容易。

按照Cindy Li的做法(如图6-3所示)，设置一个特定颜色的调色板，然后用调色板确定和创建站点的基本色彩。Web网站的徽标或任何其他商业要素也应该作为关键要素考虑。

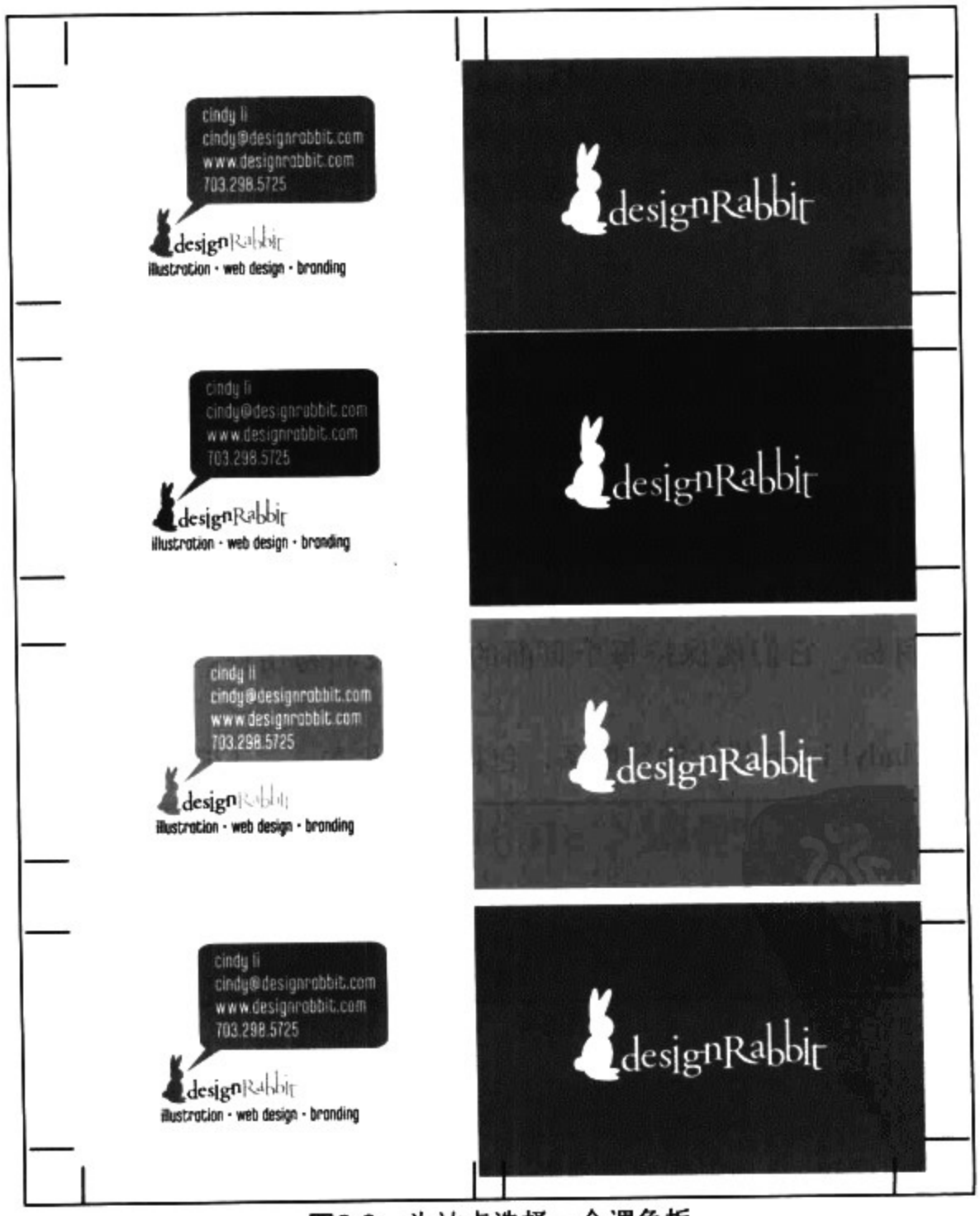


图6-3　为站点选择一个调色板

6.4 创建站点

一旦确定了基本计划，第一步就是要创建图形，以便了解图形的尺寸和所占空间。

可以用Adobe Illustrator这样的矢量图工具创建图形。用该软件能调整图形的尺寸，或在不损失质量的前提下重用这些图形。在网上还能找到其他可用的软件产品。从下列两个网站可以学习如何用Photoshop和Illustrator完成布局(和其他设计工作)。

- http://grafx-design.com/phototut.html
- http://gimp.org

准备好插图后，就可以把它导入到Adobe Photoshop或其他类似工具中，为Web站点布局做最后的确认和编辑。完成布局后看看是否喜欢这个设计(如果不喜欢则应重新开始)，然后制定一个构建布局的计划。下一步就是对布局进行部署。

6.4.1 设计导航条

在设计网站导航时，主要考虑以下三点：

- 功能性——如果用户不能找到网站导航的链接，则设计是不正确的。
- 合理性——使站点导航意义明确很重要。例如，contact是一个重要术语，用于欢迎那些希望给您发email或打电话的用户。使用没有意义的或很难确定其意义的术语就会使访问和信息查询都非常困难。
- 导航属性(如按钮和导航条)——有了这些属性，用户就可以很容易地快速找到自己所需目标。它们能保持每个页面的一致性和易用性，特别是在长页面中非常有用。

图6-4是为CindyLi.com设计的导航条，包括两种状态——正常态和翻转态。

图6-4 导航条

1. 调整大小

为了在Adobe Photoshop中创建导航条，需要打开Photoshop、输入导航文本、给每个状态20px的空间。高度与最大状态的高度取齐——在这里是翻转态(带菊花和破折号的粉

红色文本)，它有额外的高度。

导航图像的宽度取决于导航条内的字符数。通过选择Image→Image Size命令，可设置图像的最后尺寸，如图6-5所示。CindyLi.com导航图像的尺寸为350px×74px。对于您自己的导航图像而言，如果字更大，则图像的宽度应更宽一点。

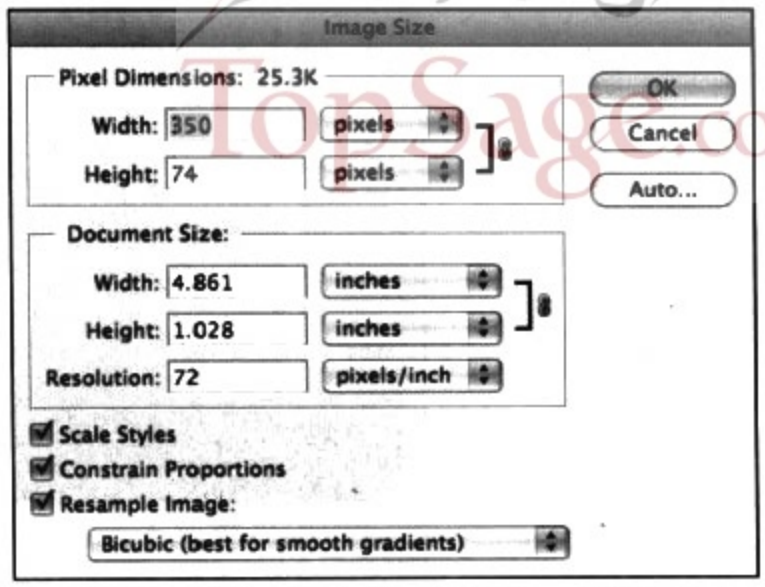


图6-5　Adobe Photoshop中的Image Size窗口

2. 调色

选择Edit→Fill命令，然后用合适的背景色填充图像窗口。可以在Contents字段区域选择Color选项(双击Photoshop工具条的“设置新图层颜色”图标)，打开Color Picker窗口(如图6-6所示)。对于CindyLi.com而言，选择的是浅蓝色，其十六进制值为#d6ebf7，RGB值为R:214、G:235、B: 247。

选择好颜色后单击OK按钮，图像就被所选颜色填充了，如图6-7所示。

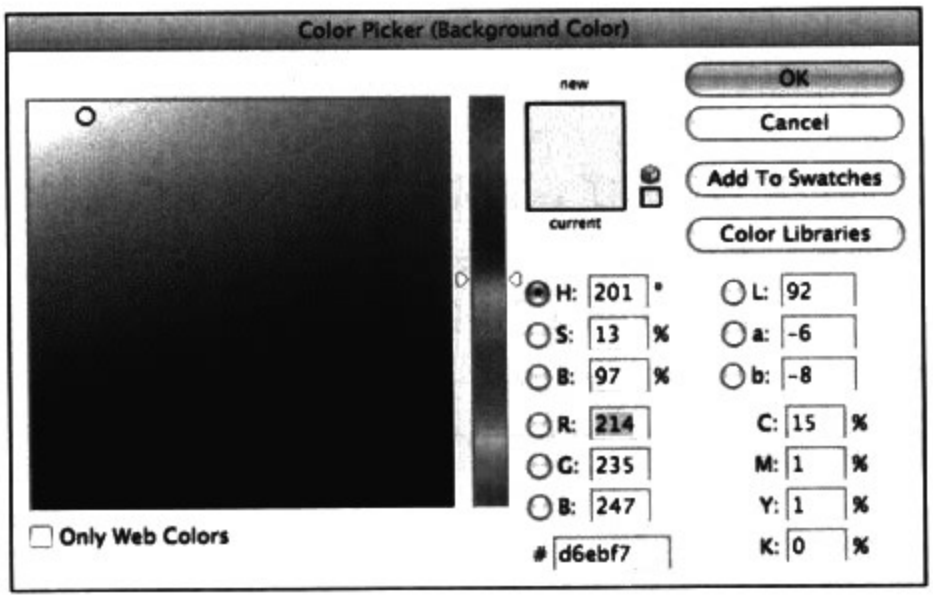


图6-6　Adobe Photoshop用于选择背景色的拾色器

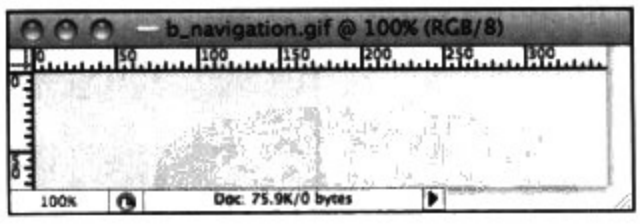


图6-7　Adobe Photoshop填充背景色示例

下一步是为导航文本选择颜色。双击Photoshop工具条的“设置新图层颜色”图标，打开Color Picker窗口。对于CindyLi.com而言，选择的是深蓝色，其十六进制值为#497690，RGB值为R:73、G:119、B: 114，如图6-8所示。由于深蓝色和背景色对比度高，则增加了文字的可读性。

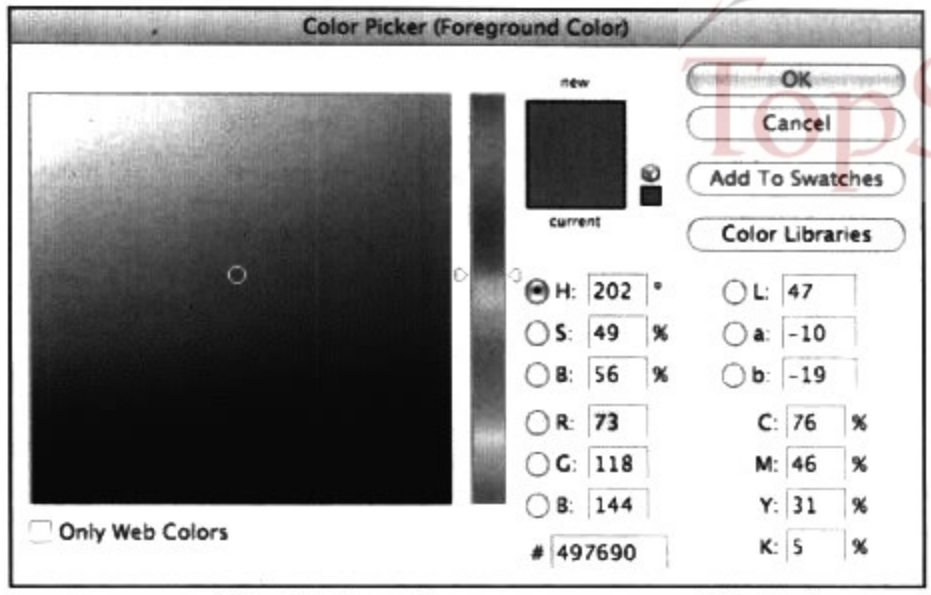


图6-8　选择了蓝色后的Adobe Photoshop拾色器窗口

3. 文本

下面设置文本的字体和字号， CindyLi.com采用24号的Cocon字体。输入导航条文字——在这个示例中是home about archive contact links。这是一个单词串，如图6-9所示。

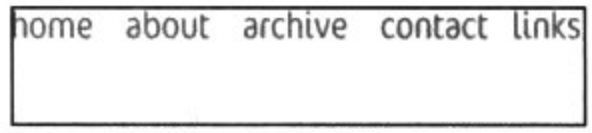


图6-9　用蓝色Cocon字体输入的导航文本

6.4.2　创建翻转图形

完成主导航标题后，接下来就该创建翻转图形了。首先在把这个文本层复制到Photoshop中，并在原来的文本标题之下创建一个20px的新文本层。

从最初的颜色方案中选择一种颜色——CindyLi.com采用的是暗粉红色(如图6-10所示)，其十六进制值为#ad0066，RGB值为R:173、G:0、B: 102。

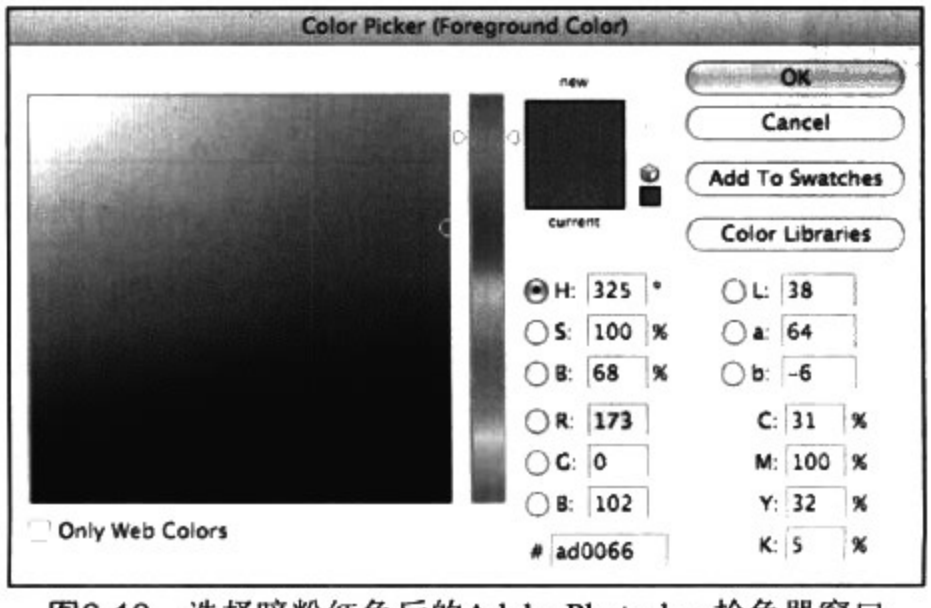


图6-10　选择暗粉红色后的Adobe Photoshop拾色器窗口

新文本使用刚选的颜色，如图6-11所示。

为了强调导航的翻转状态和已选择状态，CindyLi.com用另外的鲜花图案来体现。用Illustrator可画出自己所要的图形，然后导入Phtotoshop中作为站点采用的图像。图6-12就是最终的导航条。

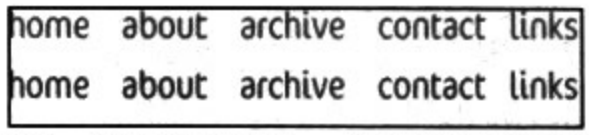


图6-11　以暗粉红色Cocon字体书写的导航条

图6-12　最终的导航图像

6.4.3　设置导航标记和CSS

完成导航图形后，就该对导航链接进行控制了，下面是相应的CSS标记。

```
<ul id="nav">
   <li id="home"><a href="http://www.cindyli.com/index.php" title="This
link takes you back to the homepage">home</a></li>
   <li id="about"><a href="http://www.cindyli.com/index.php/site/about/"
title="Find out a bit more about this weblog and myself">about</a></li>
<li id="archive"><a href="http://www.cindyli.com/index.php/site/
archives/" title="Browse the archive">archive</a></li>
<li id="contact"><a href="http://www.cindyli.com/index.php/site/
contact/" title="Contact information">contact</a></li>
<li id="links"><a href="http://www.cindyli.com/index.php/site/links/"
title="List of websites that are cool">links</a></li>
</ul>
```

用一个无序列表表示导航，然后隐藏文本以便能看到导航条中的图像(虽然文本被隐藏，但对屏幕阅读器仍可见)。

下面的CSS代码将设置导航元素的高度和宽度：

```
ul#nav {
  width:350px;
  height:36px;
  padding:0 0 0 5px;
  margin-bottom:19px;
}
```

下列代码把所有li元素设置为块级元素，这样导航元素就显示在一个水平行上。这段代码还将采用一个负的极限缩进值，对每个列表项的文本按右对齐进行缩进。

```
ul#nav li {
```

```
    padding:0;
    margin:0;
    display:block;
    float:left;
    text-indent:-9999px;
}
```

所有按钮的高度为37px，这在CSS中很容易实现。然而导航条每项文字长度不同，需要确定每个按钮的宽度并进行分别设置。表6-1列出了每个按钮所需宽度。您可以用Photoshop的度量工具自己确定这些宽度。

表6-1　每个按钮所需宽度

按　钮	标　题	宽　度
1	home	60px
2	about	70px
3	archive	85px
4	contact	80px
5	links	55px

确定了宽度值之后，可用下列代码把整个导航菜单置于每个锚项之后：

```
ul#nav li a {
  border:0;
  display:block;
  text-decoration:none;
  background:transparent url(../images/b_navigation.gif) no-repeat;
}
```

为了用CSS的宽度和高度属性确定每个导航菜单项的背景图像位置，必须针对导航菜单内的每个列表项使用一个ID属性。第一个导航菜单选项home(如图6-13所示)可用下列CSS代码进行设置：

```
li#home a {
  width:60px;
  height:37px;
}
```

菜单选项about(如图6-14所示)可用下列CSS代码进行设置：

```
li#about a {
 width:70px;
 height:37px;
}
```

下列CSS代码用于设置导航菜单选项archive(如图6-15所示)：

```
li#archive a {
  width:85px;
  height:37px;
}
```

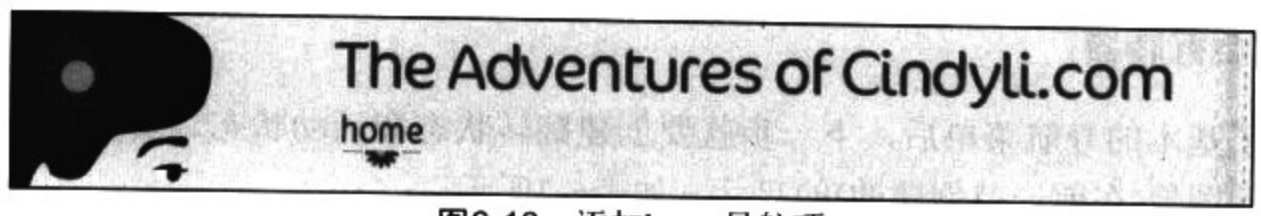


图6-13　添加home导航项

图6-14　添加about导航项

图6-15　添加achive导航项

菜单选项contact(如图6-16所示)可用下列CSS代码进行设置：

```
li#contact a {
  width:80px;
  height:37px;
}
```

下列CSS代码用于设置导航菜单选项links(如图6-17所示)：

```
li#links a {
  width:55px;
  height:37px;
}
```

图6-16　添加contact导航项

图6-17　添加links导航项

6.4.4　整合翻转器

设置了基本的导航菜单后，下一步就要创建翻转状态和活动状态。要实现这一点，需要计算移动图像(在前一节创建的)的尺寸，如表6-2所示。

表6-2　移动图像的尺寸

按　钮	标　题	翻转器状态尺寸	说　明
1	home	0px，-37px	与所创建的高度为37px的空间吻合
2	about	-60px，-37px	这个按钮移动了-60px，因为第一个按钮宽度为60px
3	archive	-130px，-37px	这个按钮的宽度取决于前两个按钮的宽度(60px+70px=130px)
4	contact	-215px，-37px	这个按钮的宽度取决于前三个按钮的宽度(60px+70px+85px= 215px)
5	links	-295px，-037px	这个按钮的宽度取决于前四个按钮的宽度(60px+70px+85px+80px=295px)

有了这些尺寸后，创建翻转效果的代码就应该如下所示：

```
/* Main navigation "hover" */

li#home a:hover, li#home a:focus {
  background-position:0px -37px;
}
li#about a:hover, li#about a:focus {
  background-position:-60px -37px;
}
li#archive a:hover, li#archive a:focus {
  background-position: -130px -37px;
}
li#contact a:hover, li#contact a:focus {
  background-position: -215px -37px;
}
li#links a:hover, li#links a:focus {
```

```
  background-position: -295px -37px;
}
```

为了创建visited状态，我们用前一示例中同样的值进行编码(第二个值，即背景的纵向位置除外)。我们需要把图形按纵向放置，因此把值从-37px改为0。代码如下所示：

```
li#about a:link, li#about a:visited {
background-position:-60px 0px;
}
li#archive a:link, li#archive a:visited {
background-position: -130px 0px;
}
li#contact a:link, li#contact a:visited {
background-position: -215px 0px;
}
li#links a:link, li#links a:visited {
background-position: -295px 0px;
}
```

为了在用户正在浏览该页面时自动触发翻转或悬停状态，需要为每一部分适当附加一个body元素。例如，about页面的body元素应该是这样的：

```
<body id="about-page">
```

最后一部分CSS代码用于改变每个页面的导航，这是用带两个ID选择符的后代选择符实现的。(这个选择符的优先级很高，它重写了导航的翻转或悬停效果，防止翻转效果出现在该页面上。)

```
/* Main navigation "active"
----------------------------------------------------------*/
body#home-page ul#nav li#home a {
  background-position: 0px -37px;
}
body#about-page ul#nav li#about a {
background-position: -60px -37px;
}
body#archive-page ul#nav li#archive a {
  background-position: -130px -37px;
}
body#contact-page ul#nav li#contact a {
  background-position: -215px -37px;
}
body#links-page ul#nav li#links a {
  background-position: -295px -37px;
}
```

6.5 设置说话框

对于Cindy的新Web站点，其主要内容部分引出了一个感兴趣的问题。说话框(speech bubble，类似漫画中常见的说话框，像个大气泡)需要一个纵向增长的空间。

为此创建一个题头图形，其中包含一个气球的尾巴(常见于文字框)，如图6-18所示。

图6-18 含汉字“造”的说话框顶部图形

也需要在文字框底创建一个文本卷轴底。通常引用图片比较好，特别是在试图获得正确的光源时。图6-19显示的图片就是一个体现光和阴影的很好示例。

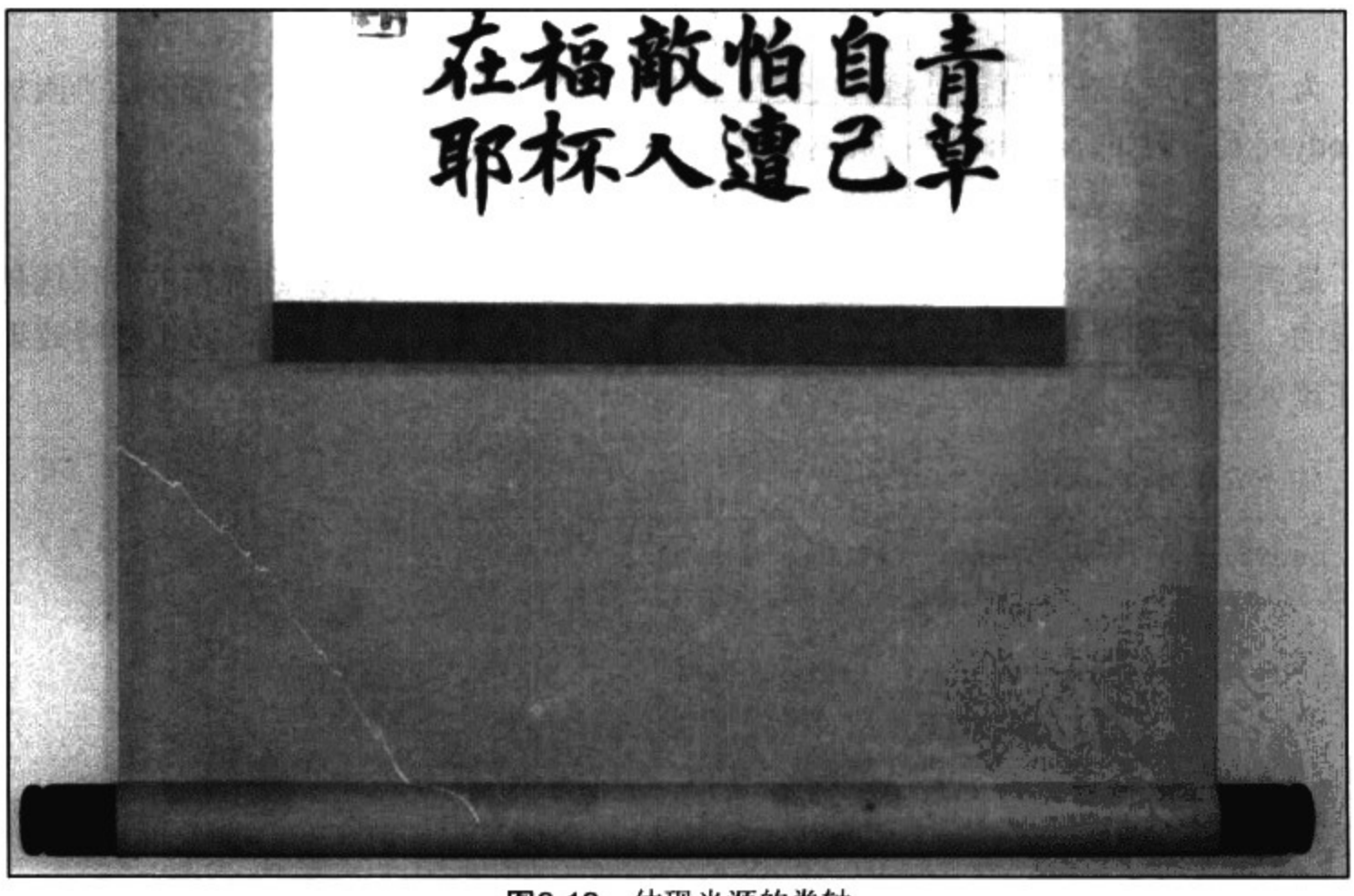


图6-19 体现光源的卷轴

在这个示例中，需要调整卷轴的边。选择一个不同的图形，如图6-20所示。

图6-20　底部卷轴图形

创建了主内容部分的底部和顶部图形之后，添加一个置于内容部分之后的图像条，以便内容扩展到不同位置时，说话框/卷轴图像以一个粘连的图像出现。

这个条图像的大小为531px×1px(如图6-21所示)。1px是重复背景条所需的高度。

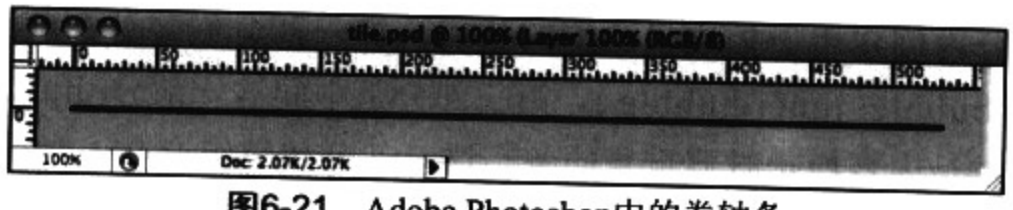


图6-21　Adobe Photoshop中的卷轴条

6.5.1　说话框编码

现在我们把精力放在说话框的编码上。第一个任务是编写标记。下面是用一系列嵌套DIV元素实现的HTML：

```
<div class="bubble">
 <div class="inner-bubble">
  <div class="inner-bubble2">
   <div>
… Blog post here…
   </div>
  </div>
 </div>
</div><!--// bubble -->
```

这四个DIV元素使图像成为粘连的，构成了说话框。

说话框的第一部分是底部部分，如图6-22所示。设置它的CSS代码如下所示：

```
/* bubble
-------------------------------------------------------*/
.bubble {
```

```
 background: url(../images/bg_bubble_bottom.gif) no-repeat bottom left;
}
```

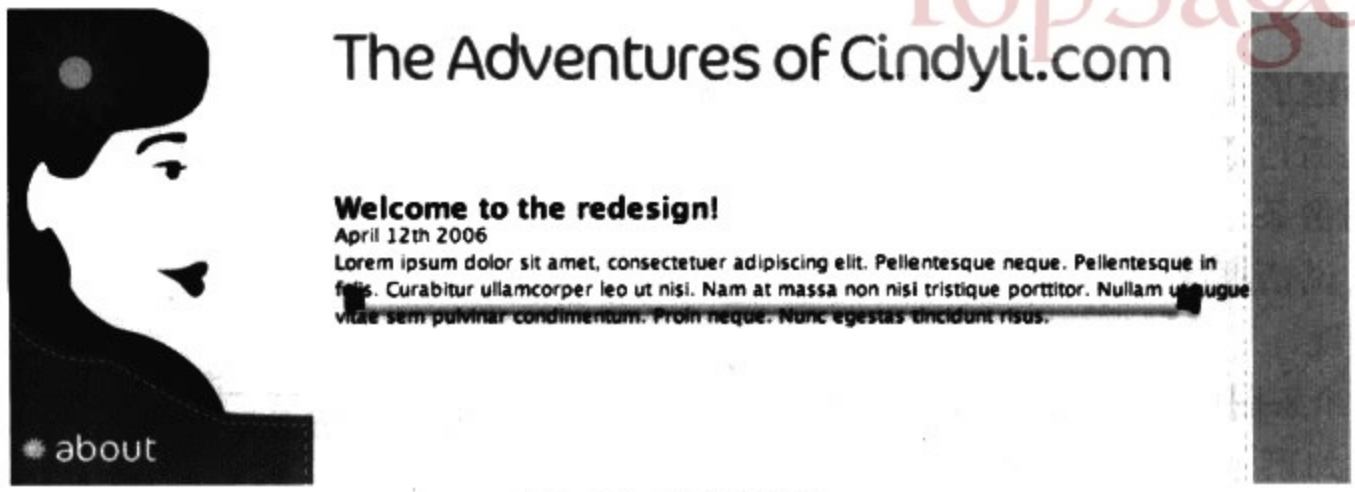


图6-22　加载底部div

需要在三部分内创建用于扩展说话框的代码。第一部分创建这个底部div。第二部分是上边这一部分(.bubble .inner-bubble.)。第三部分是中间这一部分(.bubble .inner-bubble .inner-bubble2)，这是内容增长时要重复的条。这三部分代码组合在一起后如下所示：

```
.bubble .inner-bubble {
 width: 520px;
 padding:102px 0 38px 0;
 background: url(../images/bg_bubble_top.gif) no-repeat;
}
.bubble .inner-bubble .inner-bubble2 {
 padding:1px 20px 0 40px;
 background: url(../images/bg_bubble_tile.gif) repeat-y;
}
.bubble .inner-bubble .inner-bubble2 div {
margin-top:-90px;
}
.bubble .inner-bubble .inner-bubble2 div.flickr_badge_image {
 margin-top: 0px;
}
.bubble img {
padding:4px;
margin-right:29px;
border:1px dashed #fff;
background-color:#badbef;
float: left;
}
.bubble p.centered img {
float: none;
```

```
}
.bubble ul li {
color: #666;
background: url(../images/bullet_star-pink.gif) no-repeat 0px 3px;
padding:0 0 8px 12px;
margin-top:8px;
border-bottom:1px dashed #a8cfe4;
}
```

6.5.2　再次应用这种效果

CindyLi.com Web站点在其他两个地方也应用了这种卷轴技术：在Lifetime To Do list和Design Rabbit的图形中 (如图6-23所示)。

图6-23　卷轴技术在其他地方的应用

6.5.3　添加Flickr徽章

接下来要在CindyLi.com的Web博客站点添加的是一个Flickr徽章。Flickr徽章允许从您的Flickr账户把一组图片粘贴到任一个Web页面。

首先进入Flickr徽章开发商的页面：http://flickr.com/badge.gne，如图6-24所示。这需要一个Flickr账户，不过可以申请一个免费Flickr账户，且申请过程很快。

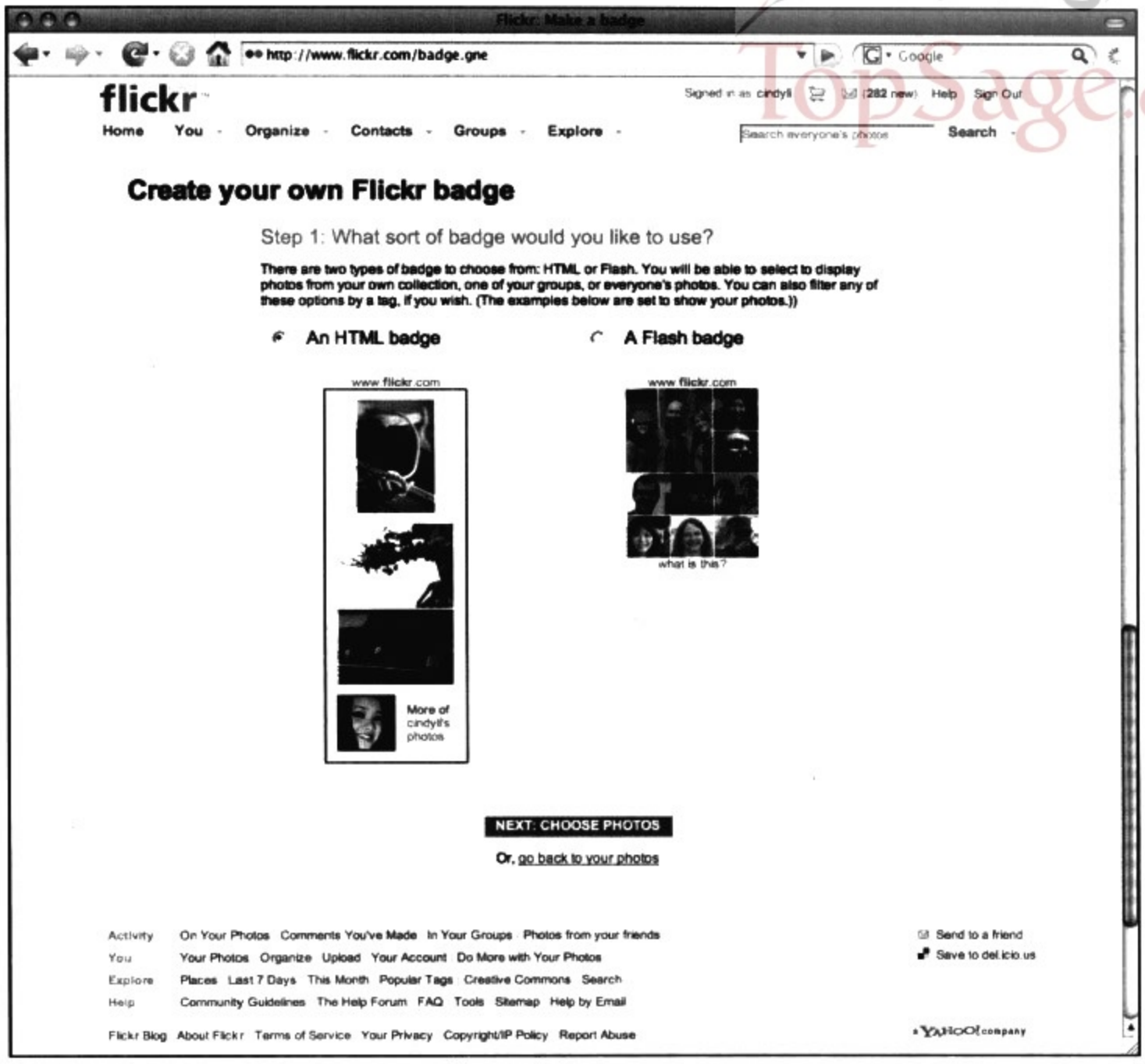


图6-24 Flickr的Web页面“create-your-own-badge”

可把Flickr徽章设置成一次可显示1、3、5、或10个图片。CindyLi.com采用的是四个图片，因此先选择三个图片的示例(如图6-25所示)，然后再进行一些修改。

如果要采用所显示的Flickr徽章示例，则采用Flickr页面框内的代码就行了，如下所示：

```
    <script type="text/javascript"
    src="http://www.flickr.com/badge_code_v2.gne?count=3&display=latest&size=
s&layout=x &source=user&user=43082001%40N00">
    </script>
```

所以Cindy需要调整这段代码中的变量才能满足要求。

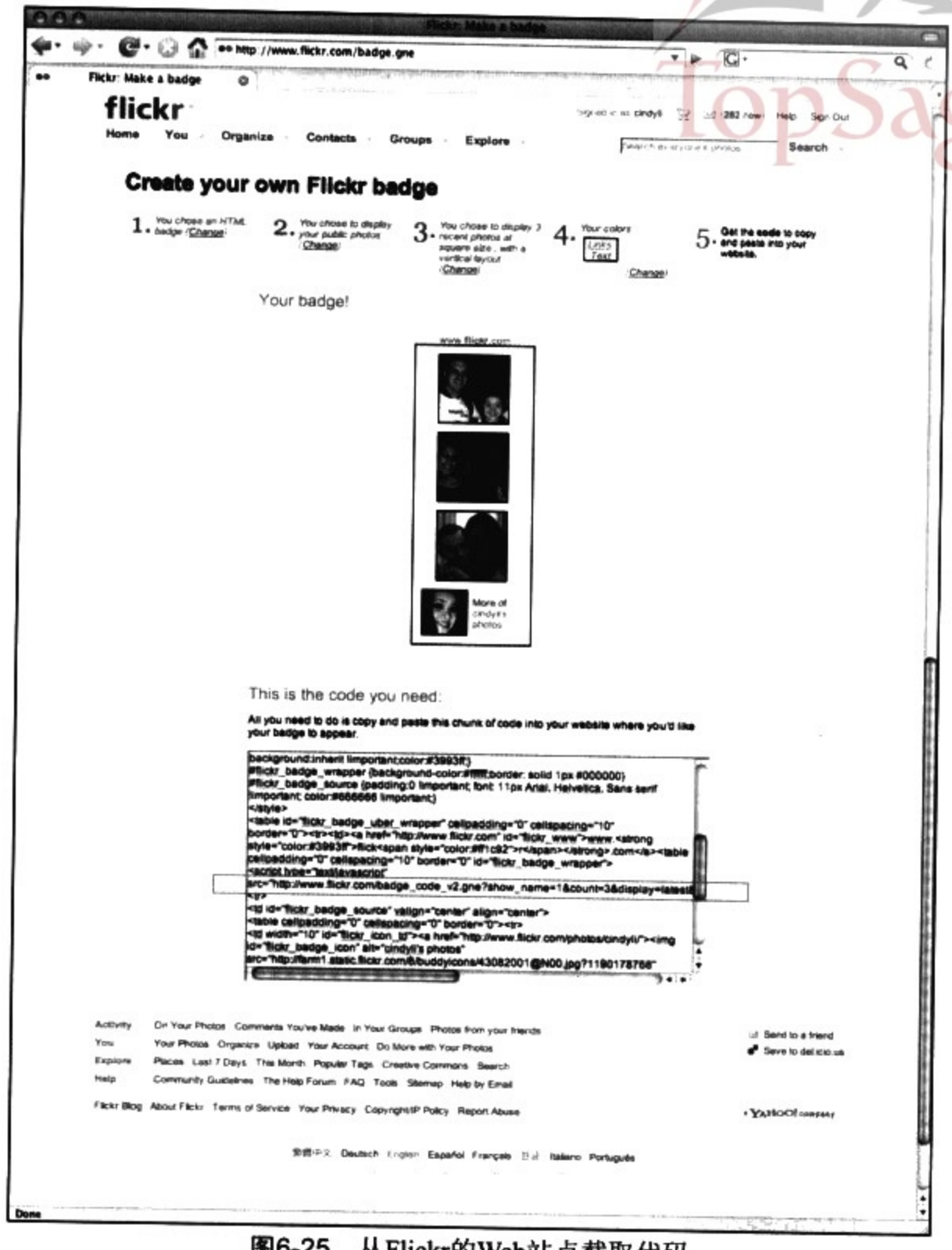


图6-25　从Flickr的Web站点截取代码

为了修改自己的变量，先把这段代码直接复制粘贴到自己的模板或Web文档中。然后复制用户号，在Cindy示例中的用户号是“43082001%40N00”。注意，&需要被改为&amp，因为这才是有效的XHTML代码。为了在自己的站点显示四个图片，需要把count=3调整为count=4：

```
<script type="text/javascript"
```

```
    src="http://www.flickr.com/badge_code_v2.gne?count=4&amp;display=latest&a
mp;size=s& amp;layout=x&amp;source=user&amp;user=43082001%40N0">
    </script>
```

然后在CSS文件中把Flickr徽章封装在一个div元素中(图6-26显示的是这段新代码的结果)：

```
<div id="flickr">
 <h2>Flickr Badge</h2>
 <script type="text/javascript"
src="http://www.flickr.com/badge_code_v2.gne?count=4&amp;...">
 </script>
</div>
```

图6-26　博客设计中没有任何样式的Flickr图片

为了把图像放在左边的粉红色卷轴区，需要一个200px×200px的div区域。把内边距设置为5px，外边距设置为10px。由于设置了外边距，因此Flickr图像被分成两行，每行两个图像。图6-27显示的是完成后的图片。

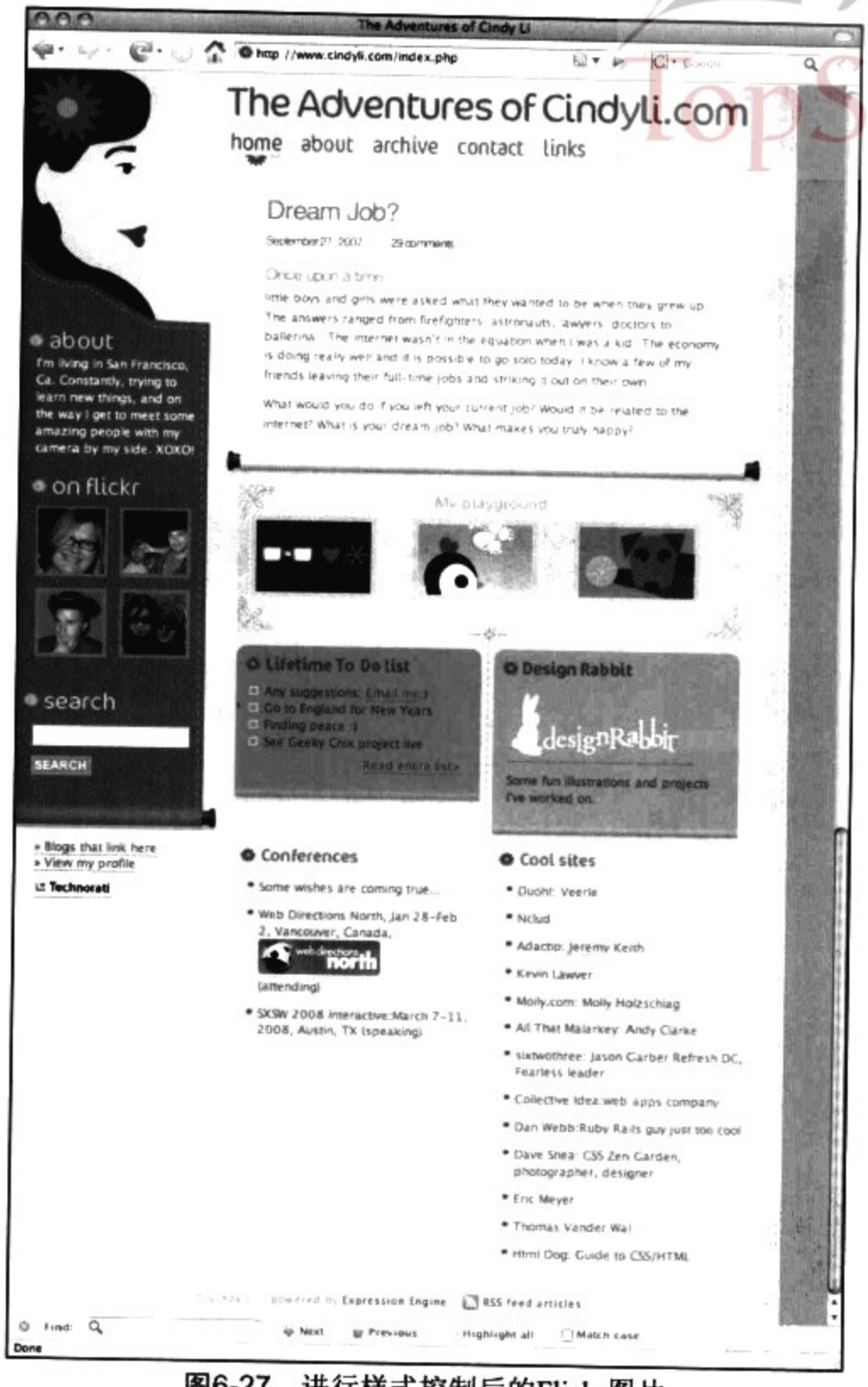


图6-27　进行样式控制后的Flickr图片

俗话说，自己的开发团队再差也比采用一个链接强。如果把自己的Web站点看着一个链条，添加一个第三方站点意味着产生了另一个链条。重要的是要使自己的站点能正常运行，必须保证第三方站点提供特定服务。对于Cindy页面的这些图片，如果Flickr徽章服务关闭了，就会变成一个空白方框，如图6-28所示。

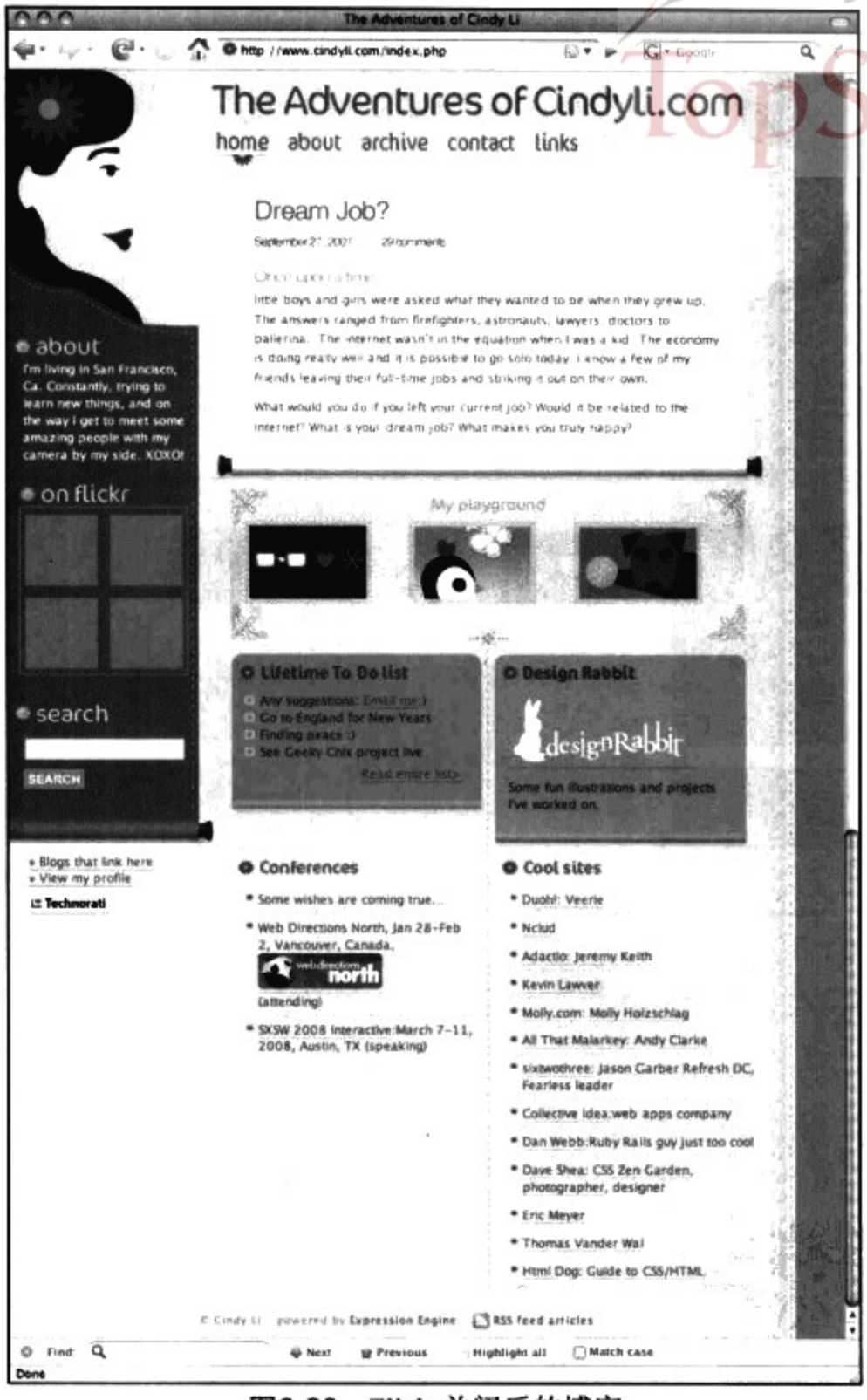


图6-28 Flickr关闭后的博客

6.6 复选框样式

CindyLi.com的Lifetime To Do List主要由是一个含复选框的无序列表(如图6-29所示)组成。

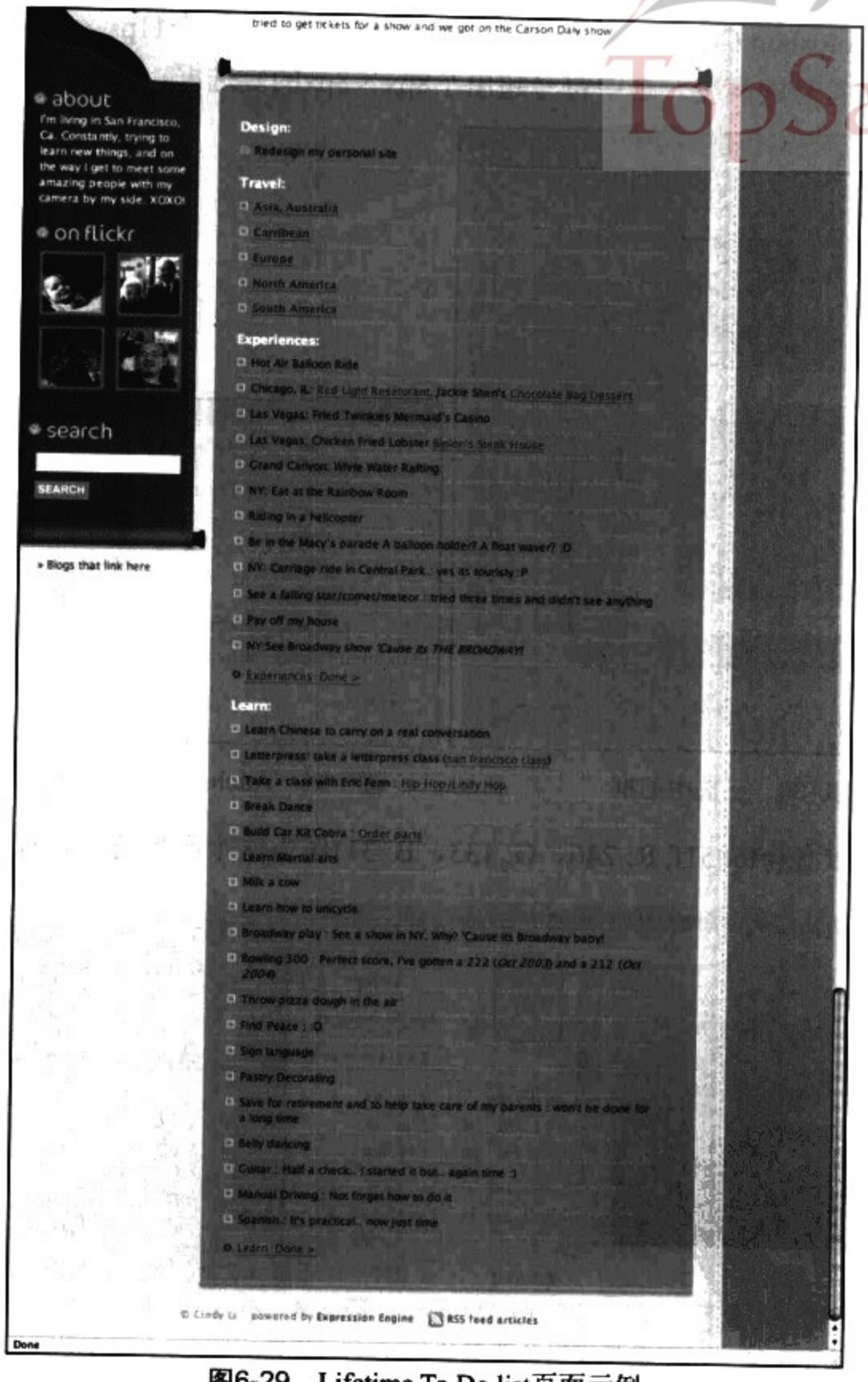


图6-29　Lifetime To Do list页面示例

为了使这个列表能正常工作，必须创建两个复选框——一个为选中、另一个为未选中。在Photoshop中看到的带选中符号的复选框如图6-30所示。

Cindy在Photoshop中按图6-31中的设置创建了自己的复选框。

在Adobe Photoshop Creative Suite (CS) 3中，按RGB创建一个11px×13px的图像。高度取决于框内选中符号的高度。选中符号超出了8px×8px的复选框。

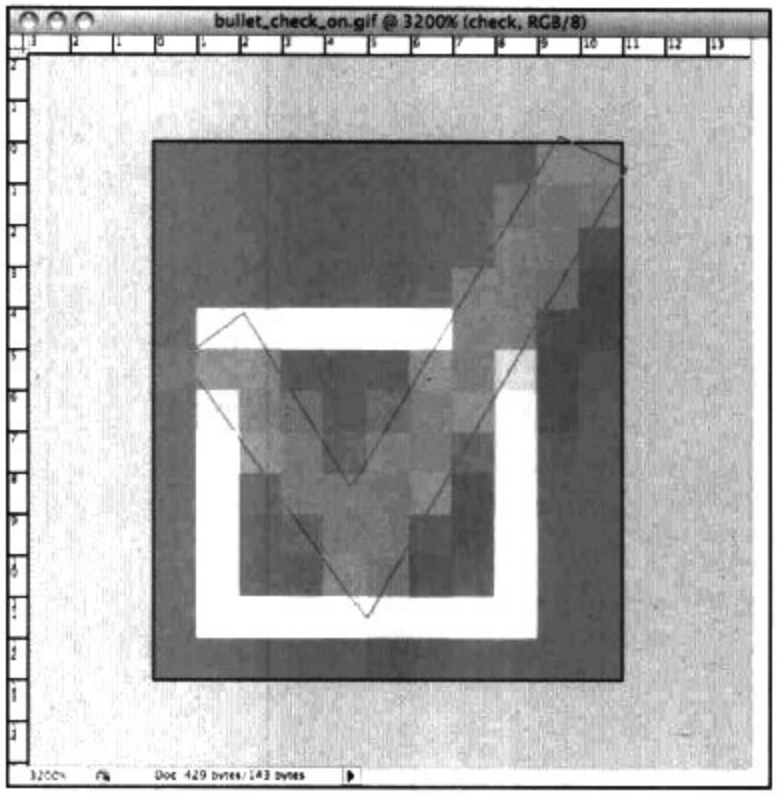


图6-30　已选中的框

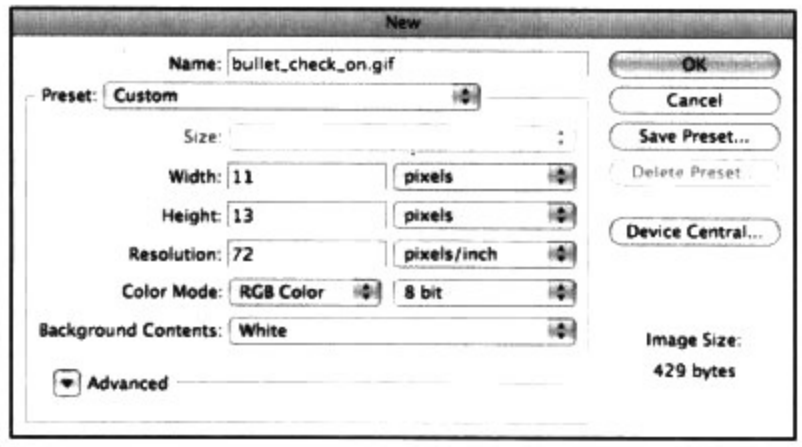


图6-31　Adobe Photoshop创建新图像对话框

Cindy 用橘红色(#f6851f, R: 246、G: 133、B: 31)作为这个图像背景，如图6-32所示。

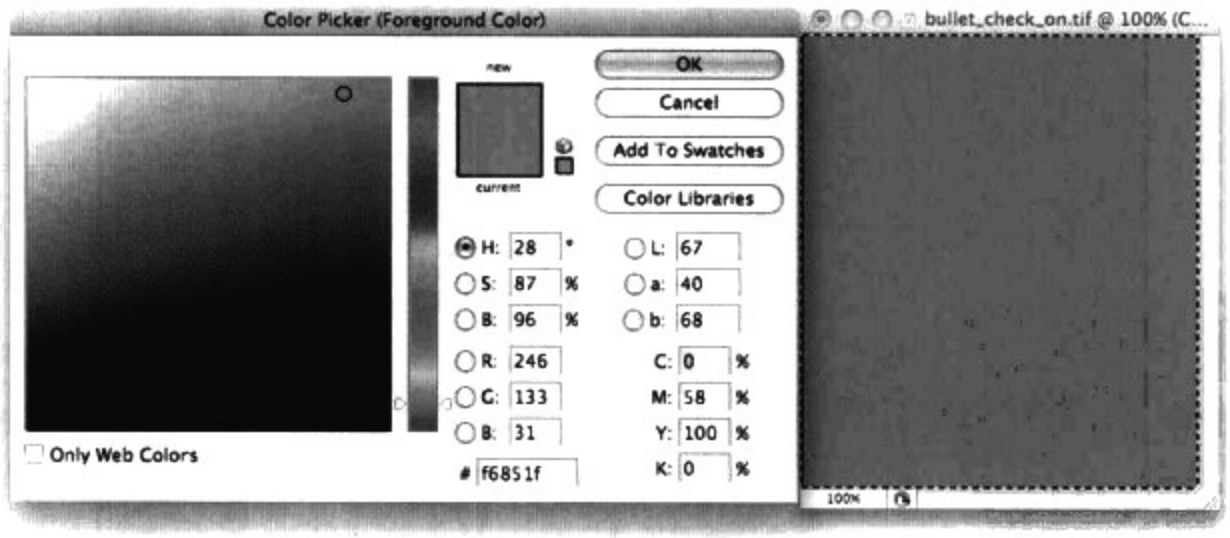


图6-32　用于选择橘红色背景的Adobe Photoshop Color Picker窗口

在Layer窗口底部，单击位于文件夹和垃圾桶图标之间的图标可创建一个新的图层，如图6-33所示。也可以选择Layers→New Layer命令来创建新的图层。

下面创建一个8px的正方形框。当用图6-34中的矩形选框工具(Rectangular Marquee Box)时，按住shift键即可创建正方形框。

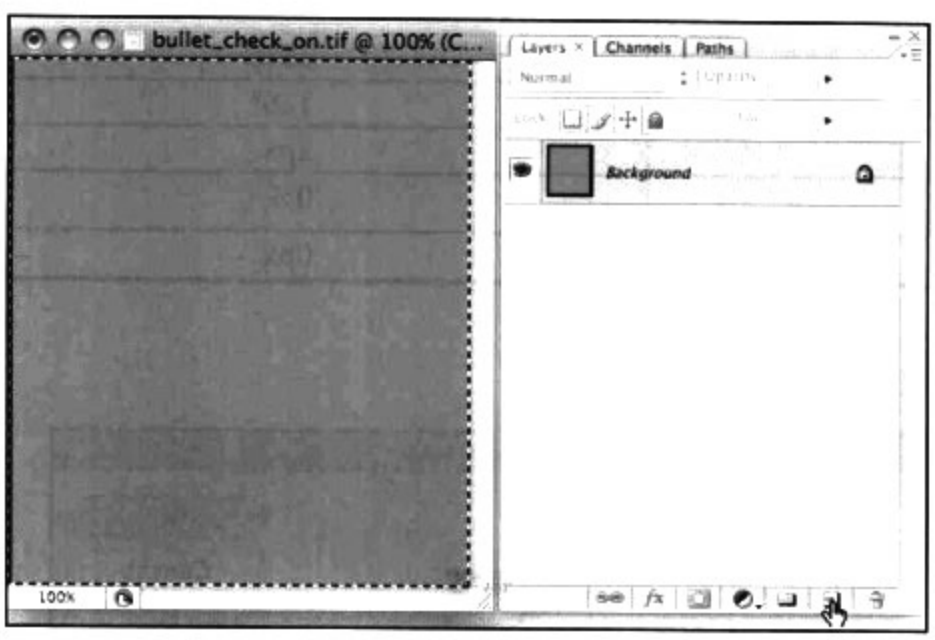


图6-33　Adobe Photoshop的图像窗口

图6-34　选中矩形选框工具后的Adobe Photoshop工具条

这个8px的白色正方形框左下角距离左边1px、距离下边1px，如图6-35所示。

为复选框创建另一个图层，并用矢量工具创建一个箭头。然后为这个箭头选择一个颜色，CindyLi.com采用的是#d1ff44，R:209、G:255、B:68，如图6-36所示。

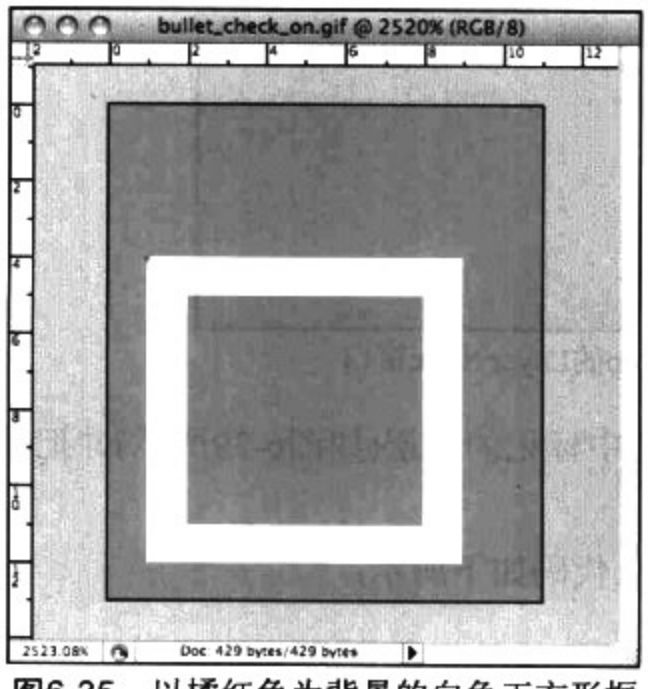


图6-35　以橘红色为背景的白色正方形框

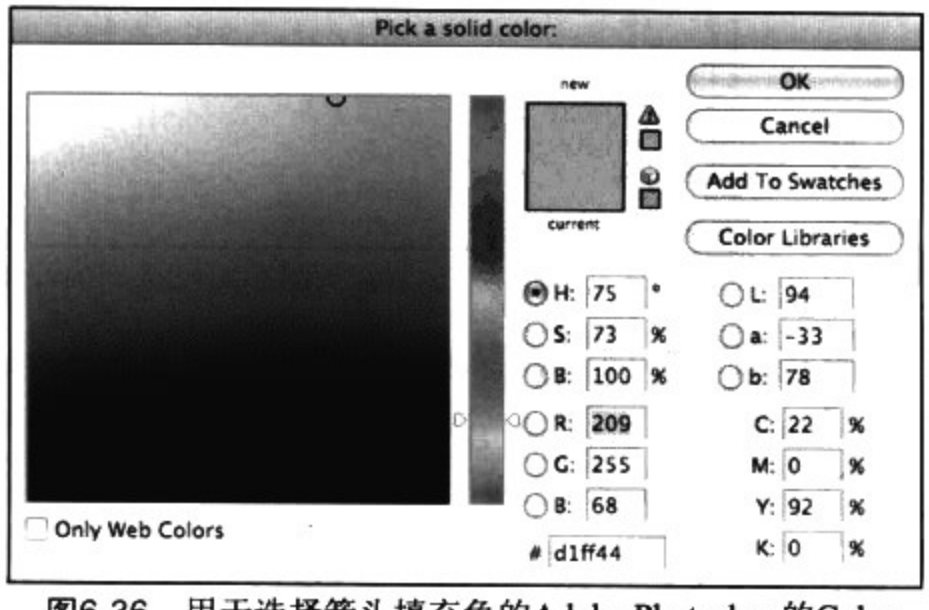


图6-36　用于选择箭头填充色的Adobe Photoshop的Color Picker窗口

按表6-3的设置为这个复选框创建一个阴影(如图6-37所示)。

表6-3 创建阴影

设 置	值
Blend mode	Multiply
Opacity	20%
Angle	120°
Distance	2px
Spread	0%
Size	0px

然后把这个文件保存为bullet_check_on.gif。

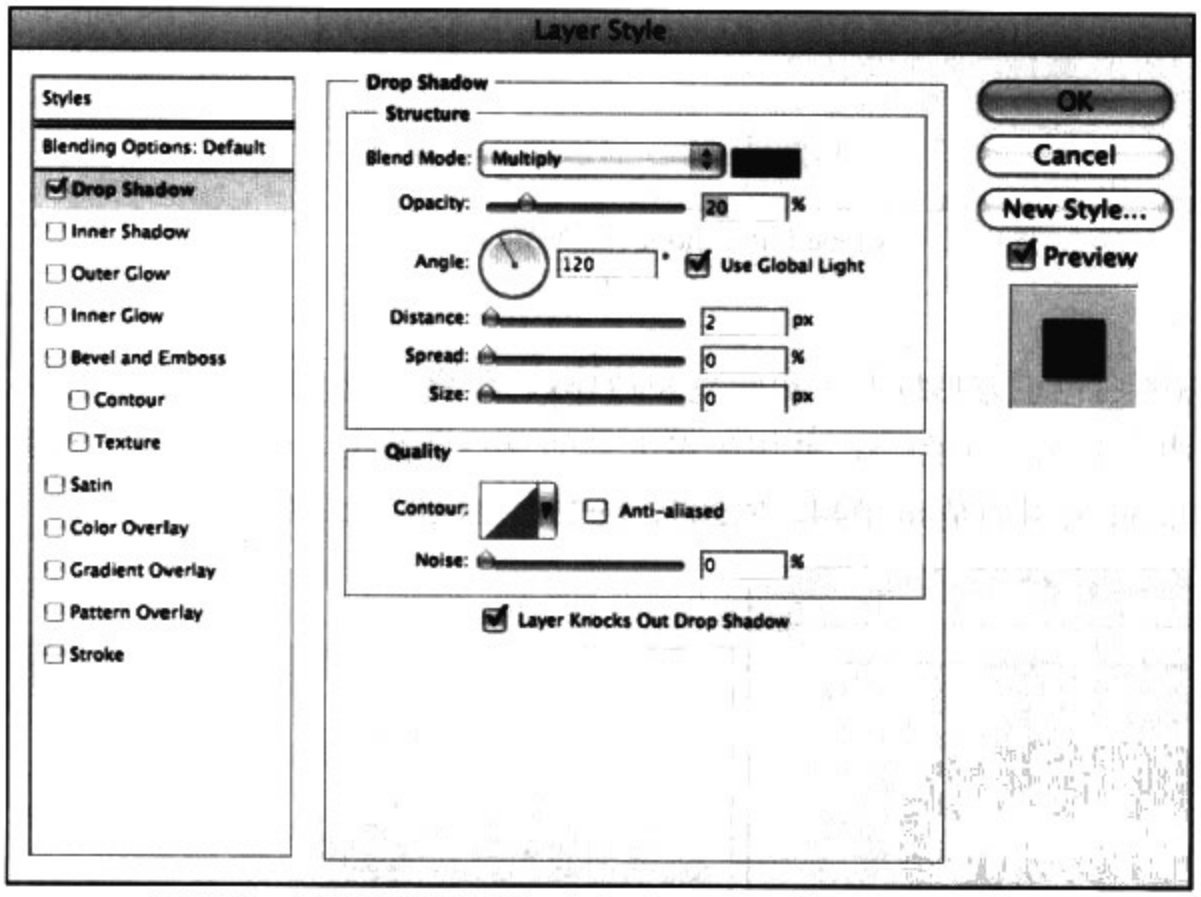


图6-37 设置箭头阴影的Adobe Photoshop的Layer Style窗口

要创建未选中的方框(如图6-38所示)，应关掉含选中标记的图层(如图6-39所示)并把图像保存为bullet_check_off.gif。

关于CindyLi.com主页上Lifetime To do List的HTML代码如下所示：

```
<h2>Lifetime To do List</h2>
<ul>
  <li class="done">Finish blog design</li>
```

```
  <li class="undone">Go to England for New Years</li>
  <li class="undone">Finding peace :) </li>
  <li class="undone">See Geeky Chix project live</li>
</ul>
```

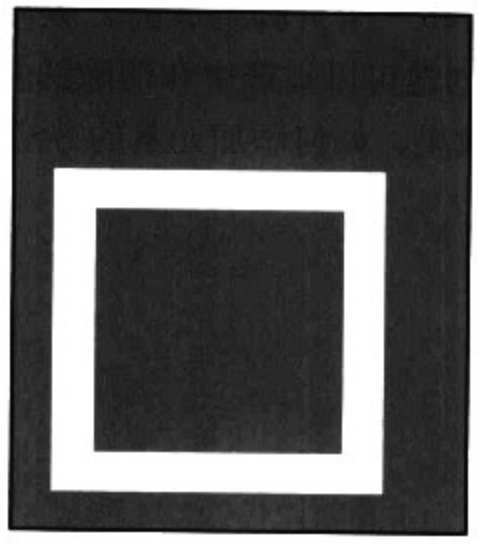



图6-38　未选中的复选框

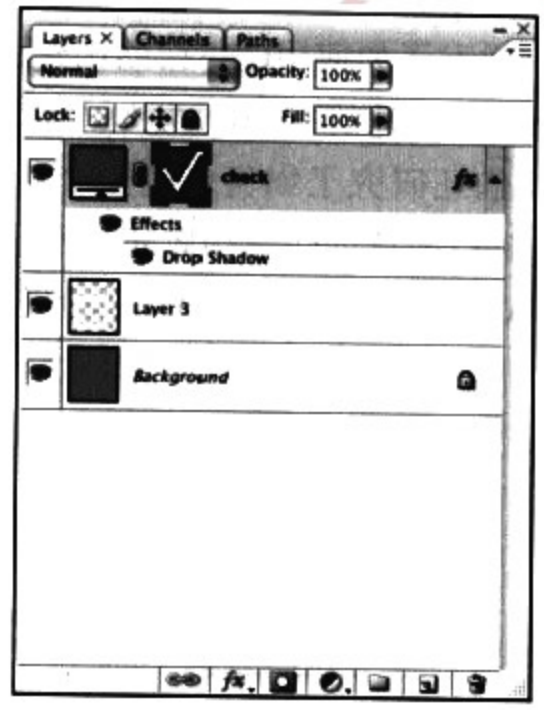


图6-39　Adobe Photoshop的图层窗口

注意，li元素是用一个class属性标记的。其值表明Lifetime To do List项是否被选中。Cindy用class属性为项目编号圆点指定合适的图形。

用CSS设计如图6-40所示的复选列表，所用代码如下所示：

```
.lifetime li {
  color:#952300;
  padding-left:15px;
  background: url(../images/bullet_check_off.gif) no-repeat;
}
.lifetime li.done {
  background: url(../images/bullet_check_on.gif) no-repeat;
}
```

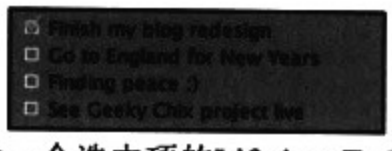


图6-40　含一个选中项的Lifetime To Do List示例

6.7 小结

本章介绍了把一个基于基本博客模板的Web站点转变为一个功能丰富的艺术作品所需的基本要素。学习了如何用CSS帮助构建这个站点并使其具有现在的功能。还学习了如何实现一个Web站点设计的布局，以及如何安装和启动一个博客、如何对博客进行定制。

另外，我们还研究了如何创建一个由一个页眉和一个页脚组成的、纵向增长的说话框；如何为Flickr徽章创建遵循W3C的代码；如何创建可以被选中和清除的复选框。在本章还演示了用CSS如何创建导航条、如何创建各种效果、如何添加元素的方法。

第7章 AIGA辛辛那提分会：HTML email模板

美国图形艺术研究协会(American Institute of Graphics Artists，AIGA)创立于1914年，其目的是把设计提高到具有专业工艺、战略性工具和富有文化表现力的高度。AIGA在全国范围内有16 000个会员和56个地方分会，是美国历史最悠久、力量最强大的设计组织之一。

像辛辛那提这样的AIGA分会是在美国全国范围内形成的，目的是增进会员之间的关系、提供工作机会、给予产品和服务一定的折扣，也会举办会议和提供一些网络机会以及建立创作室。

为了提高活动的影响，AIGA辛辛那提分会希望改进给会员的email信息，使他们及早了解协会的活动，这反过来也可以增强他们的参与意识。目前他们希望用HTML的email模板。这些定制设计的布局涵盖了他们每年的大多数活动(网络上的欢乐时光活动、学生设计的指导课程指导和设计竞赛)，但布局要非常灵活以适应突发的新事件。

7.1 HTML eamil简介

email最初用于发送纯文本内容，HTML email用于在Internet上发送图像和标记，这仅仅是一个自然的演进过程，因为商业化和消费者希望在email中有更好、更丰富的体验。

现在几乎所有流行的email客户端软件都能读取HTML email。这些客户端软件的大多

数都按默认的组成设置来创建HTML email而不是纯文本的email。很可能用户不知道他们发送的是HTML email，或不知道有“回到”发送纯文本email的选项。

由于具有更高的点击率、包含产品图像的能力、能直接链接到公司的Web站点等优势，HTML email市场比常规的纯文本email或其他广告形式更有生机(参见http://email-marketing-reports.com/basics/why.htm)。

然而缺点就是，客户端软件对HTML email的支持程度不同，也不如Safari或IE 7/Win这些Web浏览器中的email功能那样健壮。这种情形有点像20世纪90年代的浏览器大战，那时一个Web页面的代码在一个浏览器上会崩溃而在另一浏览器上能正常工作。

今天，email客户端软件处于类似的情形。在撰写本书时，email客户端软件有Apple .Mac、Google Gmail、Lotus Notes 8、Microsoft Outlook 2007和 Windows Live Hotmail，但都被Email Standards Project(ESP，在本章后面将被介绍)打上了Web标准差的标记。

通过Web标准的实现，通过Web Standards Project(WSP)和Web开发同仁的关注，浏览器开发商已把精力集中在正确实现HTML和CSS这个方向。通过同样的努力，最近形成的ESP(参见http://www.email-standards.org)鼓励生产商和开发人员做类似的改变，以实现在HTML email中对标准的完全支持。

7.2 制造模板

AIGA辛辛那提分会的模板(如图7-1所示)由设计师Joe Napier设计。

自从HTML email能把一个小型的Web页面发送到对CSS支持不太好的环境时，就开始用HTML表格控制布局。采用HTML表格的布局是今天的Web浏览器开发所不推荐的，然而在这里却是一个好的途径。为什么会这样呢？

HTML表格布局不需要HTML email客户端软件支持CSS。不论email客户端软件对CSS的支持程度如何，一个基本的HTML表格布局能把email内容更好地呈现在要求的位置。

这并不是说可以完全忽略CSS。对于该工程的HTML email模板，将用CSS定义版式、颜色，并用内、外边距确定元素的准确位置。

7.2.1 打印设计

首先要做的是打印设计。先用铅笔和尺子画出构成整个布局的表格单元。对表格的行数和列数要有个粗略的估计。设计人员利用这种方法对HTML表格结构有个大致的了解，如图7-2所示。

图7-1　HTML email模板

7.2.2　创建HTML表格布局

对HTML表格布局有大致了解之后，下一步就是在Photoshop中进行设计。打开切片工具画出表格单元。切片工具(如图7-3所示)可在一个Photoshop文件内定义几个图形的边界。

由于有快照功能，对单元格进行排列对齐就非常容易。最初所画形状的线条可作为新画切片(如图7-4所示)宽度和高度的基准。

注意：

HTML模板的上面部分没有被显示。这是因为这些图像(主图形和 AIGA Cincinnati标题)将占用整个表格宽度。所以在这一步先不管这些图形切片。通过单独输出这些图像并在代码中添加两个新的表格行可把它们引入HTML表格布局。

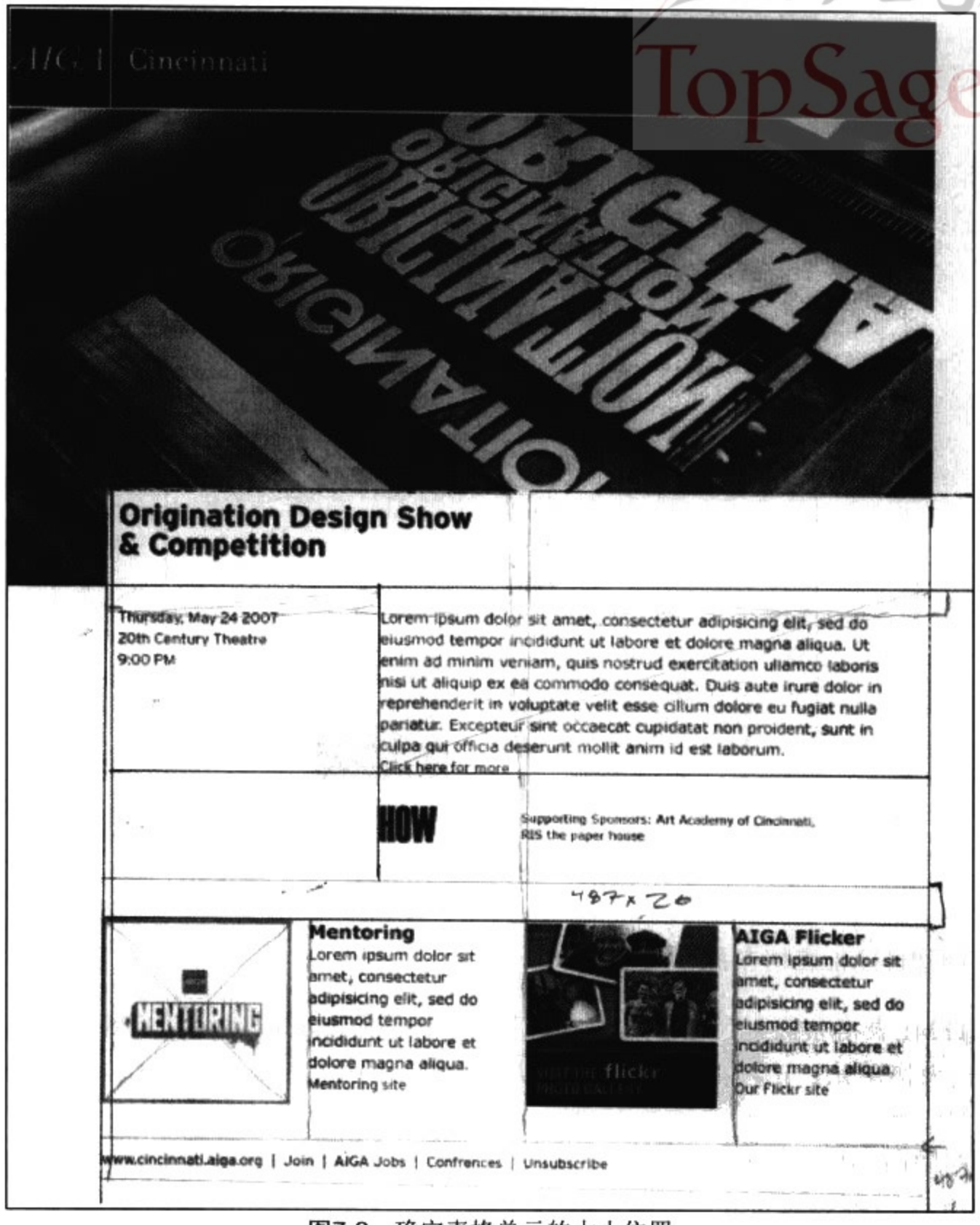


图7-2　确定表格单元的大小位置

制订了表格单元后，用切片调色板检查每个单元，并确保得到每个切片的输出。为了选择每个切片，需要用到工具Slice Select Tool(切片选择工具)。要在Photoshop中使用这个工具，应先把光标移到Select工具上并按住鼠标左键就可以使Slice菜单出现，然后选择工具Slice Select Tool(这与刀刃旁边的小箭头光标Slice Tool工具是不同的)。工具Slice Select

Tool占据了工具条中原来的Slice Tool的位置，如图7-5所示。

图7-3　在Photoshop工具箱选择Slice Tool

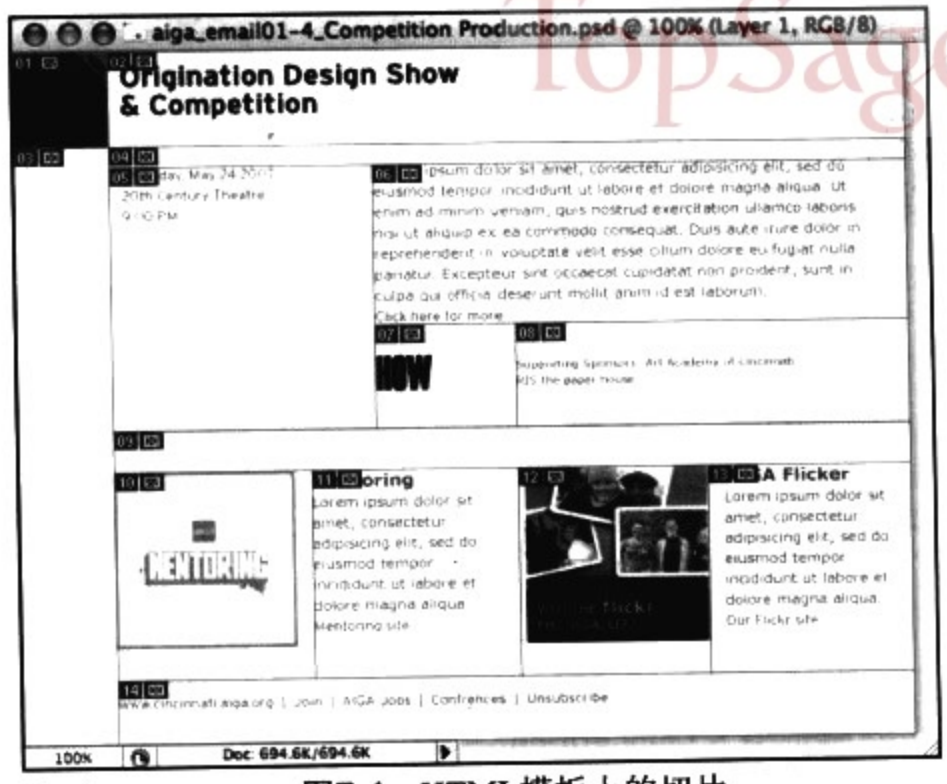


图7-4　HTML模板上的切片

在工具Slice Select Tool变为可用之后，选择左上角的第一个切片。为了调出Slice选项(如图7-6所示)，如果是PC则单击鼠标右键，如果用的是Macintosh则应按住Control键并单击鼠标左键。

我希望把第一个切片按图像导出，且赋予惟一的名称。在这里以header_tidbit为该图像的名称，如图7-7所示。

按图像方式对切片继续进行设置。但是，对于要填充文本的区域，则要将Slice Type设置为No Image，如图7-8所示。

对所有切片进行设置之后，选择File→Save for Web & Devices命令，弹出如7-9所示的Save for Web & Devices对话框。

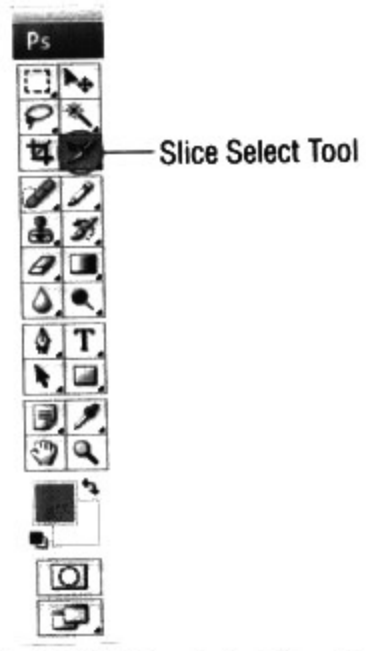


图7-5　工具Slice Select Tool的图标在左上角有一个小箭头光标

利用左上角的Slice Select工具，可以检查要转化为图像的每个切片，并对图像输出设置进行微调。

图7-6　选择Edit Slice Options命令

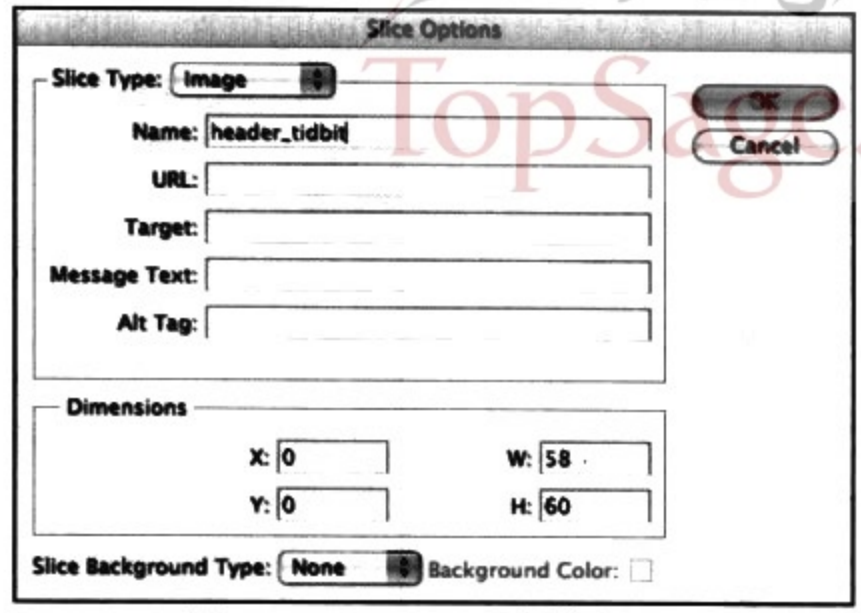


图7-7　Slice Options对话框

图7-8　设置Photoshop，使之输出文本而不是图像

说明：

对于典型的Web传输，成片区域同色的图像采用GIF文件，而类似于实际照片的图像(如设计的主图像)则按高品质的JPEG输出。

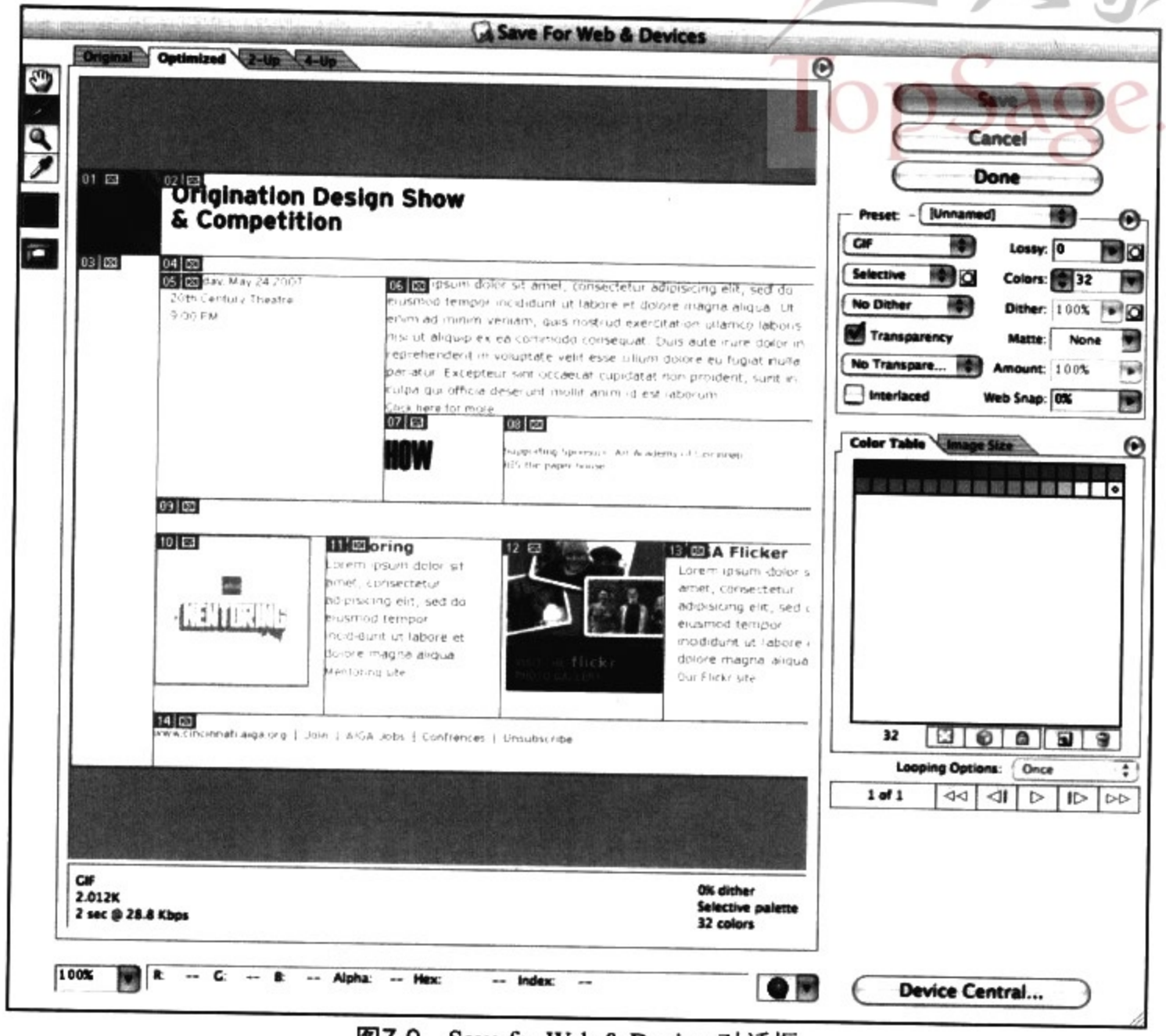


图7-9　Save for Web & Devices对话框

标记上了所有图像后，单击Save for Web & Devices对话框的Save按钮，弹出Save Optimized As对话框，如图7-10所示。

从下拉菜单中选择HTML and Images格式选项，然后单击Save按钮。这样就将Photoshop导出的所有图像自动转换为HTML表格，形成HTML email模板的基础，如图7-11所示。

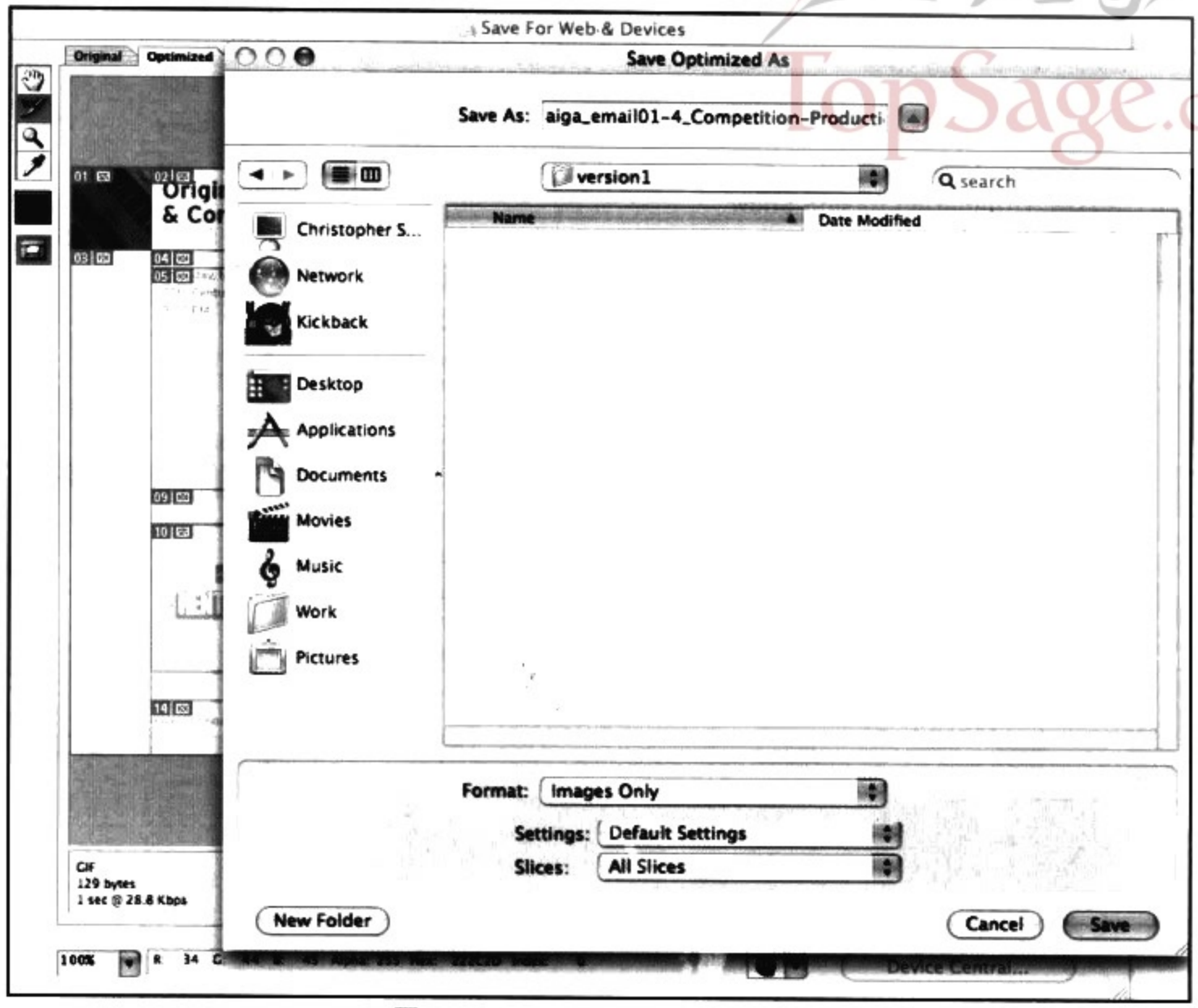


图7-10 Save Optimized As对话框

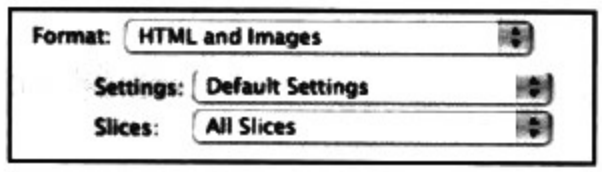


图7-11 设置导出格式

下面是导出的HTML，其结果如图7-12所示。

```
<body bgcolor="#FFFFFF" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
<!-- ImageReady Slices (aiga_email01-4_Competition Production.psd) -->
<table id="Table_01" width="545" height="436" border="0" cellpadding="0"
cellspacing="0">
 <tr>
  <td>
```

```
      <img src="images/header_tidbit.gif" width="58" height="60" alt="">
    </td>
      <td colspan="5">
       <img src="images/header_txt.gif" width="487" height="60" alt=""></td>
    </tr>
    <tr>
     <td width="58" height="375" rowspan="6" bgcolor="#FFFFFF">
      <img src="images/spacer.gif" width="58" height="375" alt=""></td>
     <td width="487" height="13" colspan="5">
      <img src="images/spacer.gif" width="487" height="13" alt=""></td>
    </tr>
    <tr>
     <td width="161" height="164" colspan="2" rowspan="2">Text goes here
  </td>
     <td width="326" height="100" colspan="3">Text goes here</td>
    </tr>
    <tr>
     <td>
       <img src="images/aiga_email01-4_Competition-Production_07.gif"
width="85" height="64" alt=""></td>
     <td width="241" height="64" colspan="2">Text goes here</td>
    </tr>
    <tr>
     <td width="487" height="26" colspan="5" bgcolor="#FFFFFF">
      <img src="images/spacer.gif" width="487" height="26" alt=""></td>
    </tr>
    <tr>
     <td>
       <img src="images/aiga_email01-4_Competition-Production_10.gif"
width="121" height="131" alt=""></td>
     <td width="125" height="131" colspan="2">Text goes here</td>
    <td>
     <img src="images/aiga_email01-4_Competition-Production_12.gif"
width="116" height="131" alt=""></td>
     <td width="125" height="131">Text goes here</td>
    </tr>
    <tr>
     <td width="487" height="41" colspan="5">Footer navigation goes here
 </td>
    </tr>
    <tr>
     <td>
      <img src="images/spacer.gif" width="58" height="1" alt=""></td>
     <td>
      <img src="images/spacer.gif" width="121" height="1" alt=""></td>
```

```
    <td>
     <img src="images/spacer.gif" width="40" height="1" alt=""></td>
    <td>
     <img src="images/spacer.gif" width="85" height="1" alt=""></td>
    <td>
      <img src="images/spacer.gif"
width="116" height="1" alt=""></td>
    <td>
      <img src="images/spacer.gif"
width="125" height="1" alt=""></td>
    </tr>
  </table>
  <!-- End ImageReady Slices -->
  </ody>
  </html>
```

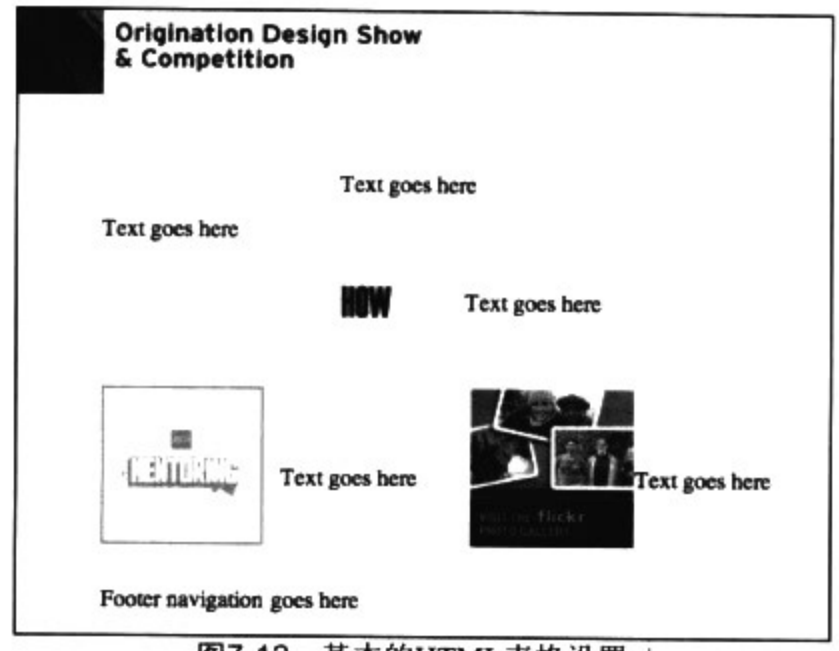


图7-12　基本的HTML表格设置

如果您还记得，原来的题头在切片功能中是被隐藏的，因为它们要占整个HTML表格的宽度。有了基本的表格结构，就可以把包含题头图形的两个表行添加到表格的顶部，如图7-13所示。

```
    <tr>
    <td colspan="6">
    <a href="http://cincinnati.aiga.
org/" style="text-decoration: none;
color:#f030a2;"><img src="images/
letterhead_agia.gif" border="0"
height="61" alt="AIGA Cincinnati"
width="545" /></a></td>
    </tr>
    <tr>
    <td colspan="6"><img src=
"images/header_designcomp_text.
gif" border="0"height="229" alt=""
width="545" /></td>
    </tr>
```

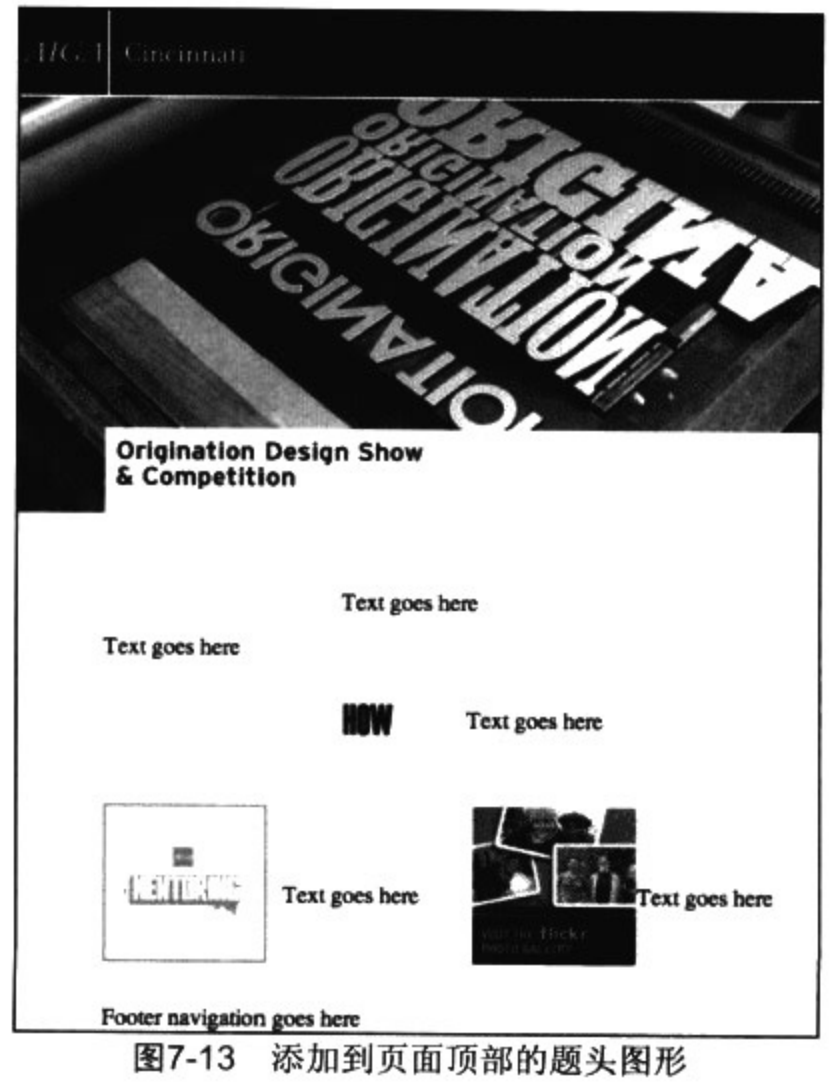


图7-13　添加到页面顶部的题头图形

接下来添加一个补白文本，其中一些已被设计人员用在最初的设计中，如图7-14所示。该新文本大大充实了设计效果。

Origination Design Show & Competition

Thursday, May 24 2007
Shaker Town, Kentucky
6:30 PM
Map

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Click here for more

Supporting Sponsors: Art Academy of Cincinnati, RIS the paper housee

Mentoring

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Mentoring site

AIGA Flickr

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Our Flickr site

www.cincinnati.aiga.org | Join | AIGA Jobs | Conferences | Unsubscribe

AIGA | the professional association for design

图7-14　补白文本使最后的产品有更好的效果

7.2.3　对设计进行调整

用HTML表格设置了模板的主要轮廓后，应用几个基本的CSS规则，使Web文档看起来与初始的设计很相像(如图7-15所示)。这实现起来也很容易。

```
h2 {
  font-size: 12px;
  font-family: Verdana, Arial, Helvetica, sans-serif;
  line-height: 1.3;
  color: #8a8b8c;
  font-weight: bold;
  padding: 0;
  margin: 0;
}
```

```
p {
  font-size: 10px;
  font-family: Verdana, Arial, Helvetica, sans-serif;
  line-height: 1.4;
  color: #8a8b8c;
  margin: 0;
}
a {
  color: #f030a2;
  text-decoration: none;
}
#eventinfo p {
  margin-left: 8px;
}
#badge1 {
  padding-rght: 28px;
}
```

注意：

这里没有采用速记CSS。我们是有意这样的，因为email客户端软件可能不够健壮，以致不能处理速记CSS。因此为了安全起见，我们把速记规则扩展成一个个单独的声明。

图7-15　应用了CSS规则的HTML模板

7.2.4　对HTML email模板的CSS规则的效果分析

下面我们对CSS逐行地进行分析，看看这些CSS对HTML email模板产生什么样的效果。

对于两个子标题(Mentoring 和AIGA Flickr)，文本字体设置为sans-serif，字体大小为12px、黑体、灰色，如图7-16所示。同时内、外边距均被设置为0。

说明：

您可能对把字型设置为像素感到惊讶。由于email客户端软件不支持em或font-size属性的关键字大小，这意味着，在某种程度上确定Web字体大小的惟一方法是用像素。这在Jeffrey Zeldman早期的一篇文章*Fear of Style Sheets*(参见http://alistapart.com/articles/fear)中有论述。

```
h2 {
  font-size: 12px;
  font-family: Verdana, Arial, Helvetica, sans-serif;
```

```
    line-height: 1.3;
    color: #8a8b8c;
    font-weight: bold;
    padding: 0;
    margin: 0;
}
```

下面分析对段落的处理，如图7-17所示。这里对子标题的处理有些变化，基于可读性考虑我们采用更浅和较小的字体。

```
p {
    font-size: 10px;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    line-height: 1.4;
    color: #8a8b8c;
    margin: 0;
}
```

图7-16　调整子标题

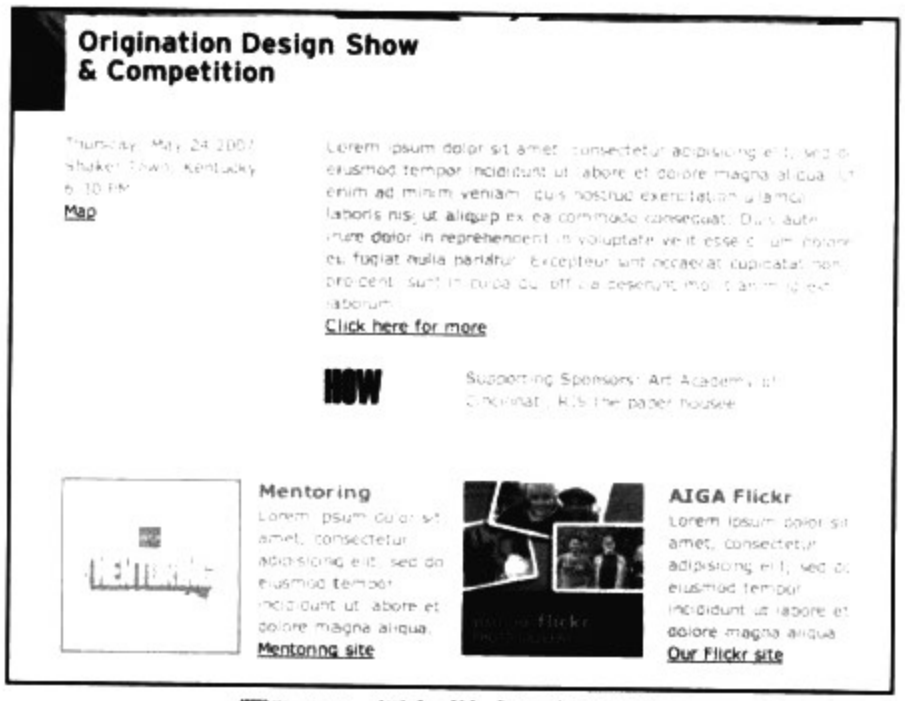


图7-17　添加样式后的段落

下一步就是设置链接的颜色。我们没有把链接设置为伪类(如在无数翻转器效果中看到的链接或悬停)，而是将所有链接设置为惟一一种颜色，即使在之前已访问过的页面，其链接的颜色也不变，因此这些链接仍然是“新”的，如图7-18所示。

```
a {
    color: #f030a2;
    text-decoraion: none;
}
```

再下一步要使两个段落文本的对齐。事件信息出现在题头的左边，Mentoring段落在右边添加了内边距，如图7-19所示。

```
#eventinfo p {
 margin-left: 8px;
}
#badge1 {
 padding-rght: 28px;
}
```

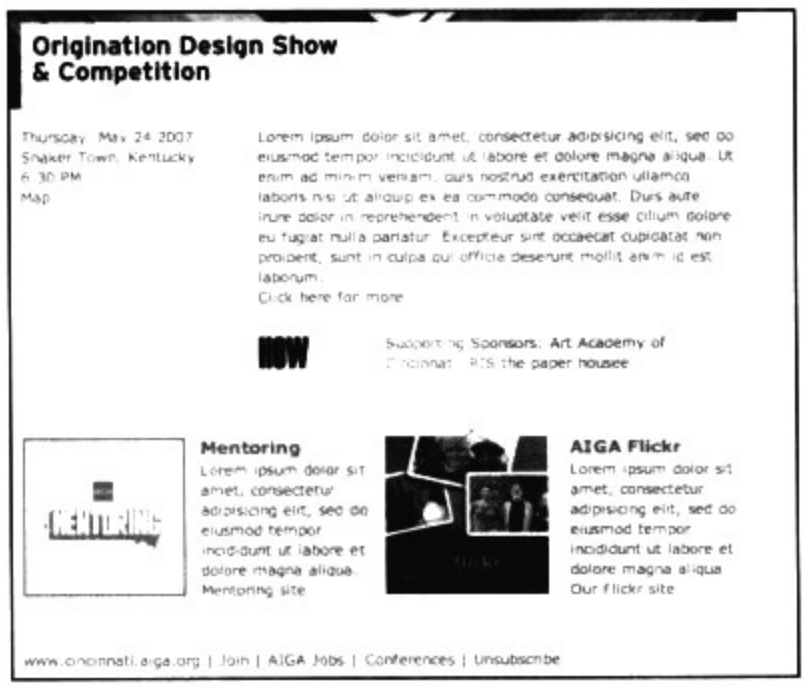


图7-18　设置链接的颜色

图7-19　调整内、外边距

7.3　嵌入样式

对于Web页面，最典型的方法是通过link元素或@import使用CSS。采用这种技术的目的是为了把表现代码和标记分离。这种分离保持了内容的干净，且使CSS的修改独立于Web文档本身。

然而在发送HTML email时，设计与内容的分离将是一场灾难，因为HTML email客户端软件可能不知道如何呈现设计中的内联或外部CSS规则。

把CSS规则从内联变为嵌入的一种方法是小心地把CSS规则复制粘贴到每个HTML元素中。下面是关于事件信息的表格单元示例：

```
    <td id="eventinfo" height="164" valign="top" rowspan="2" colspan="2"
width="161"><p style="font-size: 10px; margin: 0 0 0 8px; line-
height: 1.4; font-family: Verdana, Arial, Helvetica, sans-serif; color:
```

```
#8a8b8c;">Thursday, May 24 2007 <br />
    Shaker Town, Kentucky<br />
    6:30 PM<br />
    <a href="designcompetition.html#" style="text-decoration: none; color:
    #f030a2;">Map</a></p></td>
```

虽然这种方法可以达到目的，但是姑且不论手工嵌入的繁琐程度，这种方法也非常容易出错。值得庆幸的是有另外一种自动处理的方法。

7.4　为HTML email进行预处理

Web上的另一种服务是Premailer (参见http://code.dunae.ca/premailer.web)，这是由Alex Dunae开发的。Premailer利用典型的Web页面(带内联和外联的样式表、具有相对图像路径等)创建了一个对HTML email友好的版本。

不仅如此，Premailer(如图7-20所示)还将对CSS进行分析，确定一些流行的HTML email客户软件是否支持这些CSS。

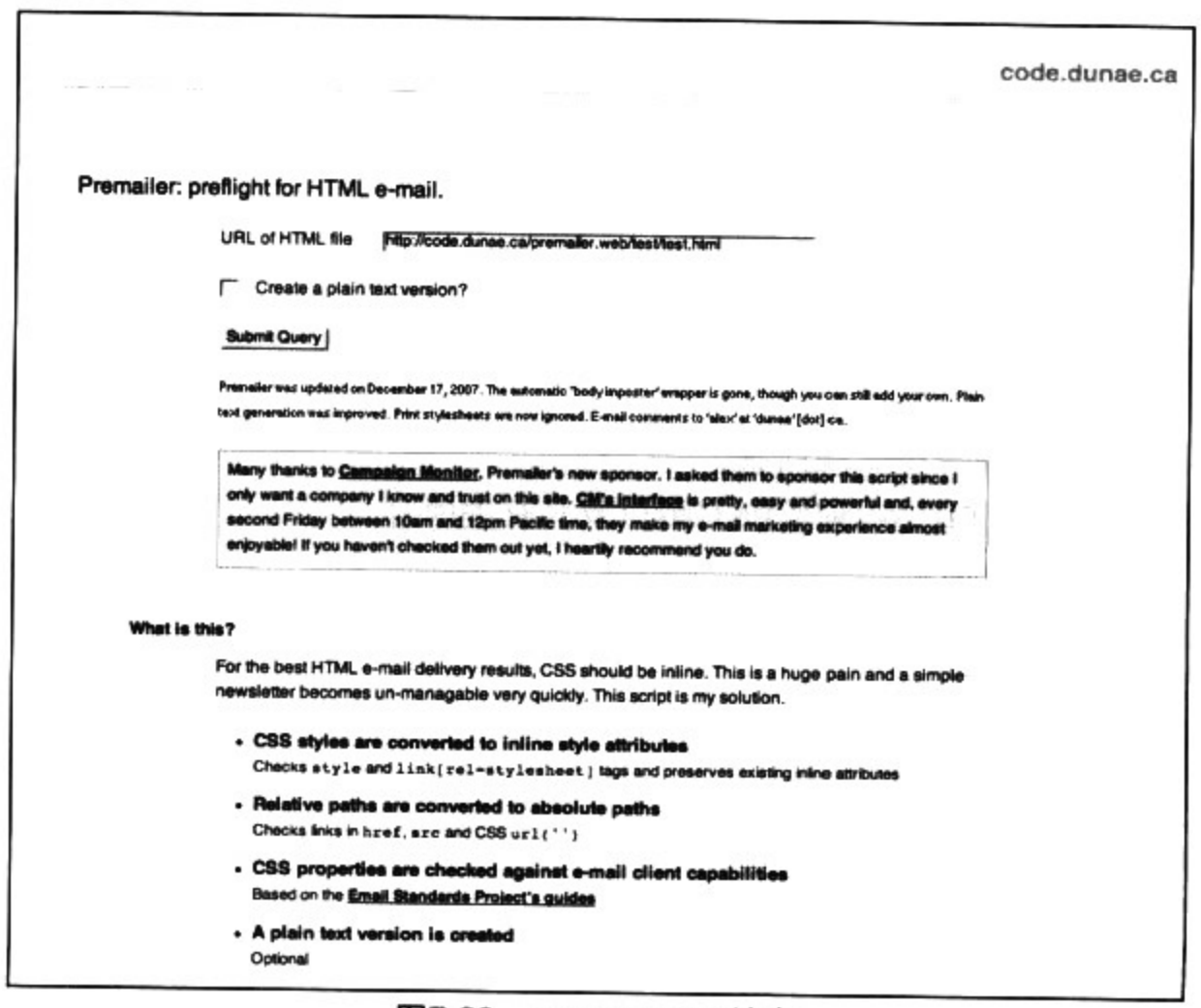


图7-20　Premailer Web站点

在把HTML email模板上传到本地的开发服务器上并提交后，这个Web服务将把CSS规则转换为嵌入的规则，并提交一个报告说明CSS规则将如何出现在不同的email客户端件中，如图7-21所示。

这个报告显示，使用margin属性是危险的。由于在本例使用这个属性的目的只是为了稍微调整一下文本的位置，以便使设计看起来更接近期望的结果，所以可以继续工作。

说明：

按照Premailer的建议，应该把margin属性完全删除，从而使CSS是安全的。如果是这样，需要创建一个更复杂的HTML表格，该表格包含额外的单元以占据CSS内、外边距的位置。

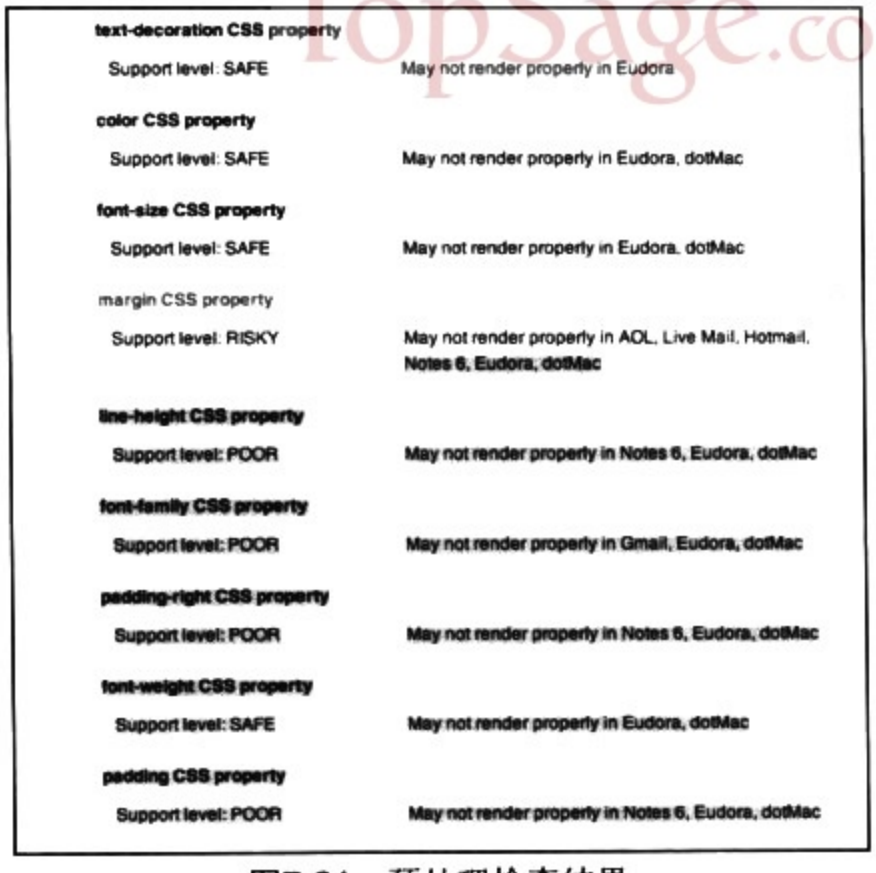


图7-21　预处理检查结果

7.5　小结

本章介绍如何用一个Web页面布局和利用Photoshop的Slice和Web工具创建一个可以使用的HTML email模板。还略微提及了如何添加精巧的CSS规则，以便其在一些非常流行的email客户端软件中呈现时不出现问题。

第8章将介绍如何在IE 6中使用PNG图形及其有关内容。

第8章

专业CSS图书网站：透明PNG图像的使用

本章我们将学习在设计一个Web站点时如何使用透明的PNG图像。我们采用的样例是一个实际的例子，即http://www.procssbook.com的主页，如图8-1所示。这个页面就是用透明PNG图像设计的。

在学习如何使用透明的PNG图像之前，先介绍为Web网站部署设计时为什么透明PNG图像会特别有用，认识到这一点很重要。

与GIF和JPEG一样，PNG很适合在Web上使用。与GIF类似，PNG特别适合显示那些颜色很少的小图像，如徽标和图标。PNG相比GIF还有几大优势。特别值得一提的是PNG支持Alpha透明。什么是Alpha透明？GIF文件只能把一个像素显示为全透明的或全不透明的：这就是所谓的二元透明。当一个图像包含Alpha层时，则图像的这些地方就是部分透明的。可以指定一个0～255之间一个透明级。图8-2是不同透明级图像的比较。

当与大多数图像和图形文件打交道时发现，PNG的压缩比要比GIF更高。与GIF相比，PNG提供了更多的透明选择方案。Alpha通道透明是可用的第一个选择。一个PNG图像还可在一个图像中提供更宽的颜色深度，这远远超过了一个标准的GIF图像。用PNG，可以用最高达48位的真彩色取代8位(或256种颜色)彩色。这提供了更丰富的颜色控制，例如，我们可以产生一个更平滑的褪色效果。因此利用PNG就可能在一个Web页面上产生一些有趣效果，比如一个半透明的背景图像和阴影。

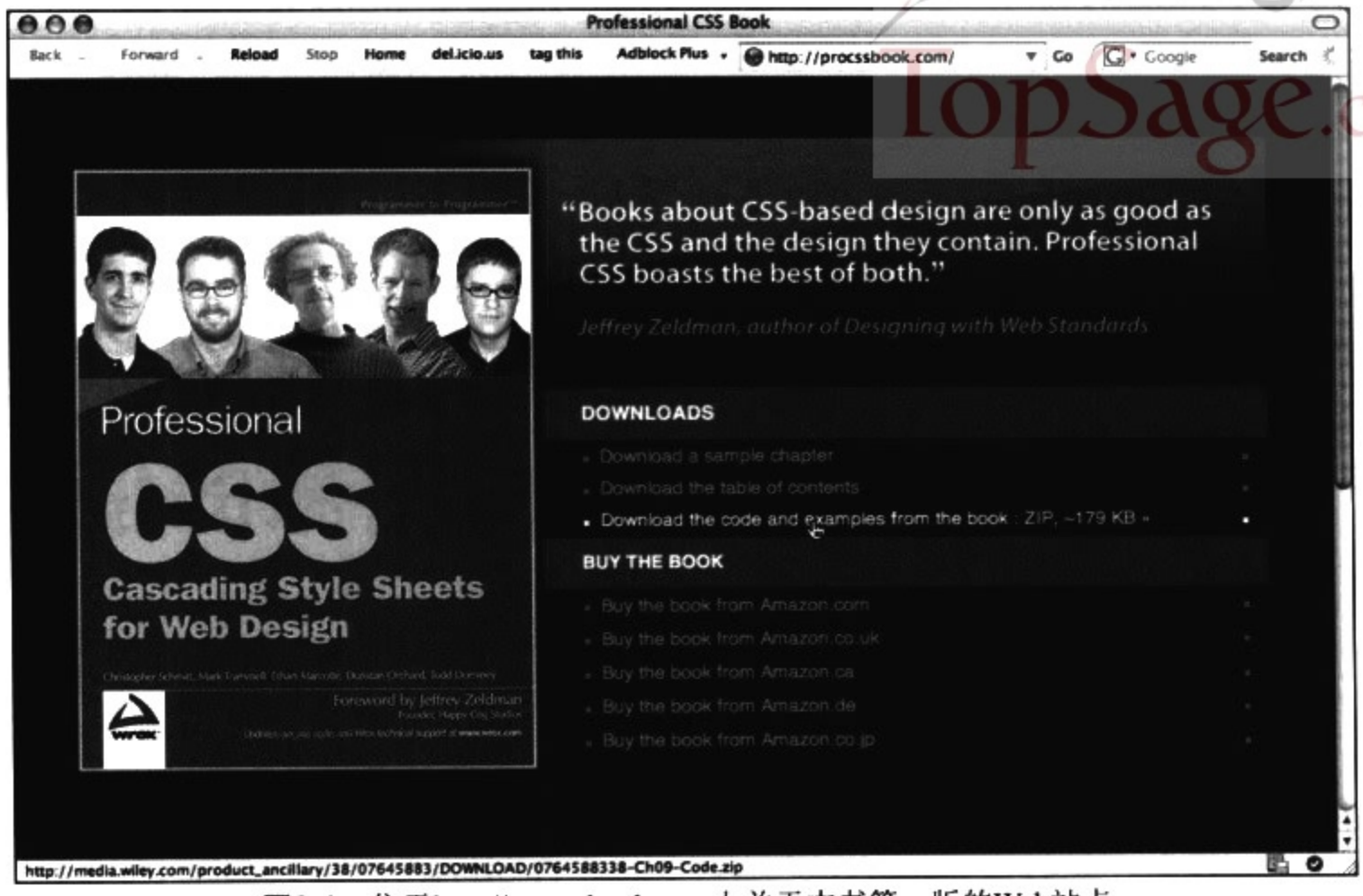


图8-1　位于http://procssbook.com上关于本书第一版的Web站点

PNG是一个很好的选择，但不否认GIF的两大优势：支持动画的能力和浏览器对GIF的普遍支持。PNG和GIF相比尽管有很多优点，但在Web设计中PNG远没有GIF流行，这主要是由于存在这样的印象：PNG没有得到浏览器的广泛支持。对PNG的这种看法是错误的。

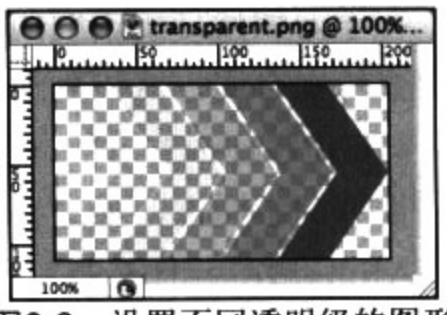


图8-2　设置不同透明级的图形

8.1　PNG和浏览器支持

虽然IE 6/Win及其以前的版本不支持PNG的Alpha透明功能，但所有流行的浏览器——Safari、Opera、Firefox、甚至IE 6的后继版本IE 7/Win——都能显示PNG图像。

说明：

所有这些现代浏览器都支持PNG的Alpha透明，还有必要为IE 6担心吗？肯定不能简单地下结论。请您亲自检查日志文件看看还有多少用户在使用IE 6。如果所占比例很小，接近0%，就不必要为IE 6担心。

虽然IE 6不能显式支持Alpha透明，但如果愿意的话可用一个迂回方法保证PNG的跨浏览器的兼容性。

8.1.1　在IE 6中使用PNG的图像过滤方案

Microsoft有大量专门的视觉滤镜和渐变工具(参见http://msdn2.microsoft.com/en-us/library/ms532847.aspx)可供IE 4以上的版本使用。在用IE浏览时，这些滤镜把各种多媒体效果(渐变消除、光效果等)应用到Web页面的图像。其中一个滤镜是AlphaImageLoader，利用这个滤镜可在IE中显示含Alpha透明的PNG图像。

创建一个div元素并嵌入一些CSS，就可在页面的HTML中使用这个滤镜了，如图8-3所示。

```
<div style="position:relative; height: 188px; width: 188px;
filter:progid:DXImageTransform.Microsoft.AlphaImageLoader
(src='images/image.png',sizingMethod='scale');"></div>
```

这里的关键属性是filter属性。虽然filter不是合法的CSS属性，但它可把AlphaImageLoader滤镜运用到括号内指定的图像。然而，由于这不是与标准兼容的代码，因此只有在需要时才应用这个属性(即只有页面在IE 6中显示时)。

图8-3　浅紫色透过了PNG图像的透明部分

利用这种方法，对现代浏览器开发人员可以构建基于丰富图像的设计，且包含Alpha透明。这些浏览器(如Safari、Firefox和IE 7)本身就支持Alpha透明的PNG图像。

8.1.2　在IE 6中使用PNG的HTC脚本方案

要在IE 6中使用PNG的图像，另一种可行的方法是采用Angus Turnbull的.htc脚本。

首先在网站TwinHelix Designs(http://www.twinhelix.com)中下载.htc脚本。HTC只是IE(因为HTC是由Microsoft创建的)使用的一种脚本语言。这种特定的脚本语言包含AlphaImageLoader滤镜，也把AlphaImageLoader滤镜应用到Web页面的所有图像。它既可在Microsoft的IIS (Internet Information Services) Web 服务器上运行，也可在Apache(这是基于开源的Web服务器)上运行。

下载该脚本后，把它上传到您的Web服务器上。

接下来创建一个空的GIF文件。这个文件中的图像为1px×1px，颜色被设置为透明(在20世纪90年代，这种图像被称为“单像素GIF图像”)。也可以从网站http://www.twinhelix.

com/css/iepngfix下载这个图像。

在.htc脚本内修改引用blank.gif文件的行，使之指向服务器上的GIF图像位置。

创建一个独立的CSS文件(命名为ie.css)，在文件内包含引用该.htc文件位置的行：

```
img {
  behavior: url(iepngfix.htc);
}
```

behavior属性把一个脚本与一些选择符(在这里是img元素)关联。因此，该CSS文件把这个.htc文件与所有图像关联，这样就可以把所希望的过滤效果应用到Web页面的每一个图像。

但我们希望只有在用IE 6浏览该页面时才加载这个CSS文件。要做到这一点，只需在页面的header部分添加下列条件注释：

```
<!--[ if lte IE 6]>
<link rel="stylesheet" type="text/css" media="screen"
href="ie.css" />
<![ endif] -->
```

IE能理解这样的条件注释。这段注释的意思是，“如果浏览器是IE 6或更早的版本，则读注释标签内的行。否则忽略这些行”。利用条件注释，可以方便地应用与IE有关的HTML或CSS。在这里，只有用IE 6浏览该页面时才加载样式表ie.css，这样只有在绝对必要时才采用与标准不兼容的CSS。

呈现内联图像的常用技术在下列情况可能失效：

- 当在元素的背景中放置图像时，图像的正常行为是遮住元素。采用这种方案，PNG图像不会遮住元素，因为设计了filter属性来实现这种效果。
- 不要与CSS sprites技术结合起来使用这种方案。要确保为这种效果使用一个且只有一个图像。
- 如果试图在一个Windows操作系统中运行多个版本的IE，可使用http://tredosoft.com/Multiple_IE所介绍的一种类似方法，条件注释可能会失效，且这个工作图像可能出现在本地系统中。但这种方案也可能一直有效而不出现问题。可用如BrowserCam(http://browsercam.com)这样的第三方测试工具进行测试。

使IE 6脚本发挥作用的主要方法是把图像从内联位置放到一个元素的背景中。

8.1.3 PNG图像的颜色问题

另一个问题是颜色修正问题。是否注意到Mac上的颜色比PC上的要浅？这是由构建这

两种系统的架构师造成的。通过在PNG图像中包含Gamma信息可解决这个问题。

这个过程大致是这样的：用数字创作软件记录呈现PNG文件中的图像所需的Gamma信息和其他必要数据。然后通过Internet发布这个PNG图像，并在各种操作系统和浏览器上显示。由于PNG文件中包含Gamma信息，浏览器或任何其他应用程序就会用合适的颜色修正值显示这个PNG图像。

问题是这些信息是不完全的，应用程序在其他人的系统上不能重新构建最初的颜色信息，也得不到正确显示颜色的正确设置。您上一次校准计算机显示器是什么时候？

当图像中包含Gamma信息时，系统就会试着以最佳方式呈现PNG图像，但这种解决问题的尝试使问题更糟。

说明：

*当然这是对问题的简单描述。有关该问题的更详细信息请参考*http://hsivonen.iki.fi/png-gamma。

最佳方案是放弃存储在PNG图像中的Gamma信息。如何实现？如果已经使用了Photoshop的Save for Web功能，则PNG设置就会剔除Gamma信息。如果没有这个软件且使用的是Mac，可以尝试使用Shealan Forshaw的Gammaslamma (参见http://www.plasticated.com/GammaSlamma-1.1.dmg)下载 .dmg文件。

8.2　使用Alpha透明

尽管存在IE 6的支持问题，但由于PNG图像提供了很多好处，所以值得我们使用。

8.2.1　更好的阴影

用PNG图像可以更容易地实现阴影。由于GIF图像只能把一种颜色设置为透明，因此需要把如阴影中的平滑梯度很好排在一起才能达到这种效果。然而，即使是图像的一个像素不可见，也会在无意中改变了设计，如图8-4和图8-5所示。

在图8-4中，本书图像的阴影与背景对齐。如果该阴影位置恰好与一个GIF图像重合，且这个GIF图像包含一个阴影，对齐就会出现错误，如图8-5所示。

由于PNG图像含透明的阴影，一个阴影只被显示一次，即使元素的像素是不完整的(对于Web大多数情况下是像素不完整的)，也不会破坏图像，如图8-6所示。

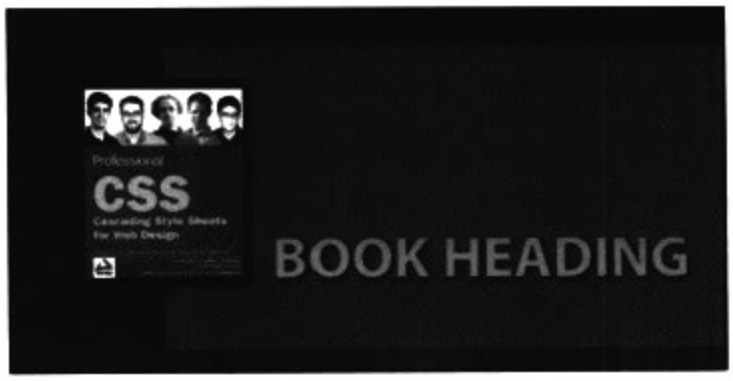


图8-4　图像的阴影与背景对齐

图8-5　出现错误对齐

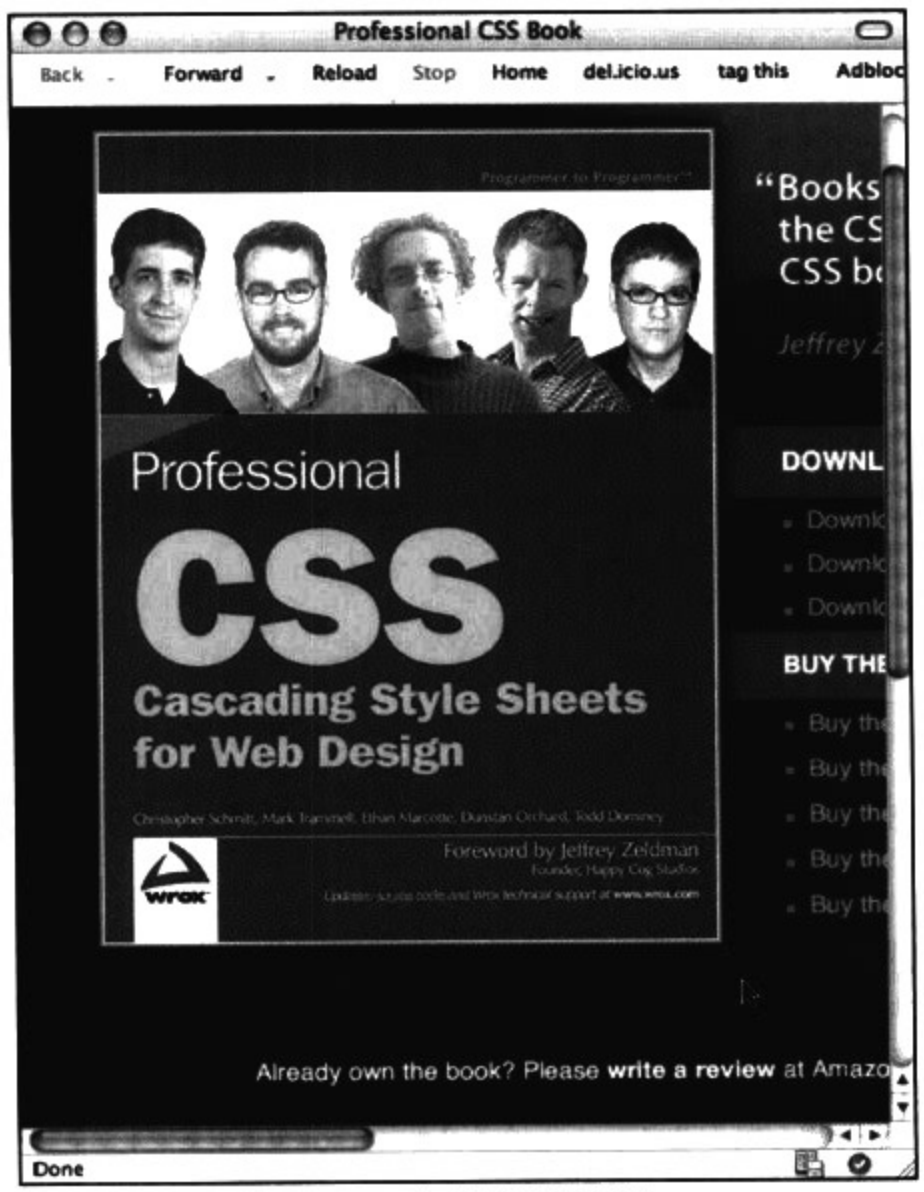


图8-6　本书封面的阴影

图8-7说明，即使是重新调整了文本大小，PNG图像的阴影也不会影响整个设计。

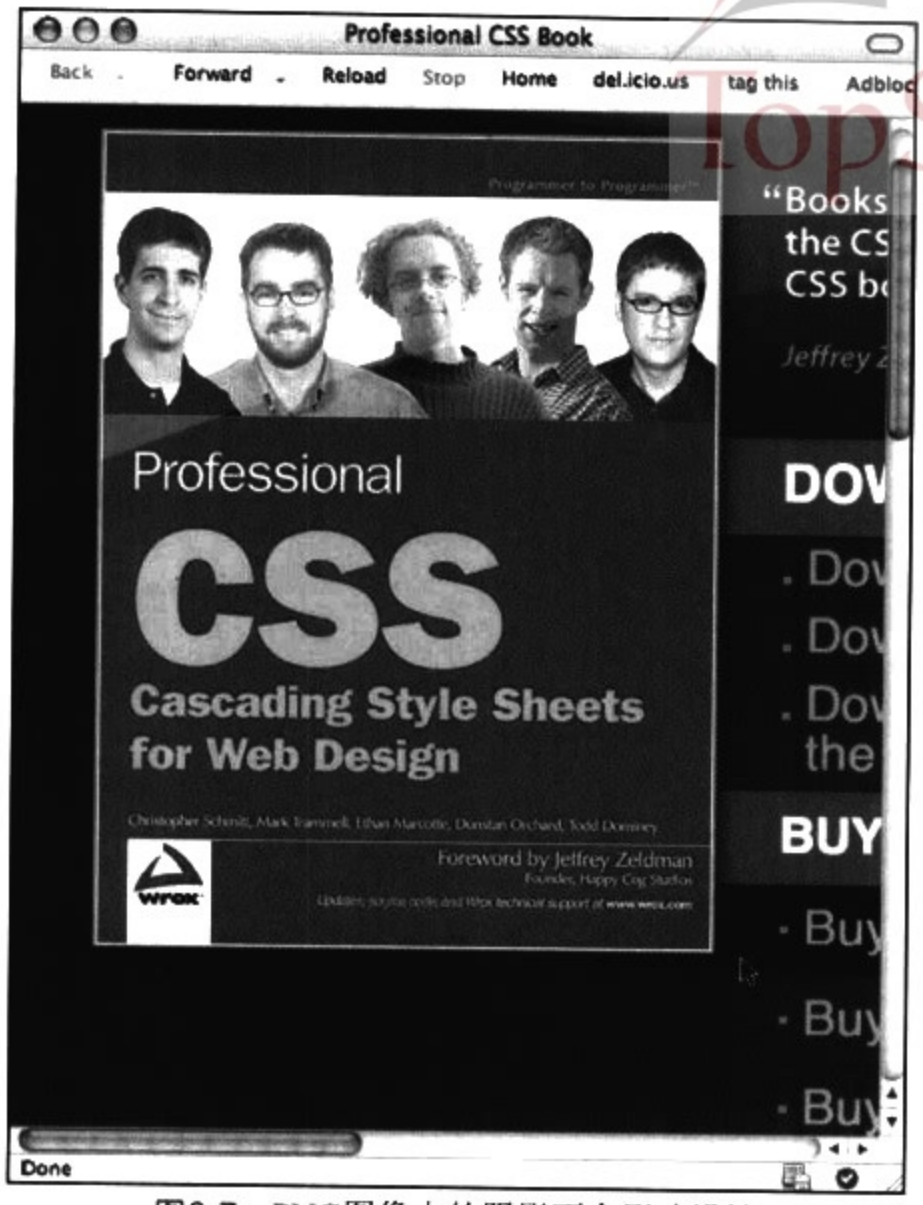


图8-7　PNG图像中的阴影不会影响设计

8.2.2　使用彩色阴影

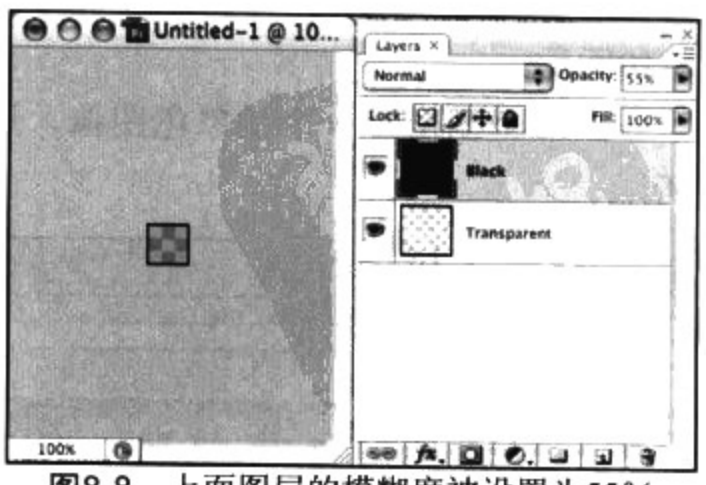


图8-8　上面图层的模糊度被设置为55%

彩色阴影是一些小的PNG图像，这些PNG图像是由设置了某个模糊度的黑色或白色组成的，可用这种颜色给背景色或图像着色。可从 http://christopherschmitt.com/2007/03/16/color-shades/ 下载一组彩色阴影，也可以自己创建。

为了创建自己的彩色阴影，在Photoshop设置一个24px×24px的图像，然后调整模糊度，如图8-8所示。

选择File→Save for Web & Devices命令以输出这个图像，并将其命名为PNG-24，如图

8-9所示。

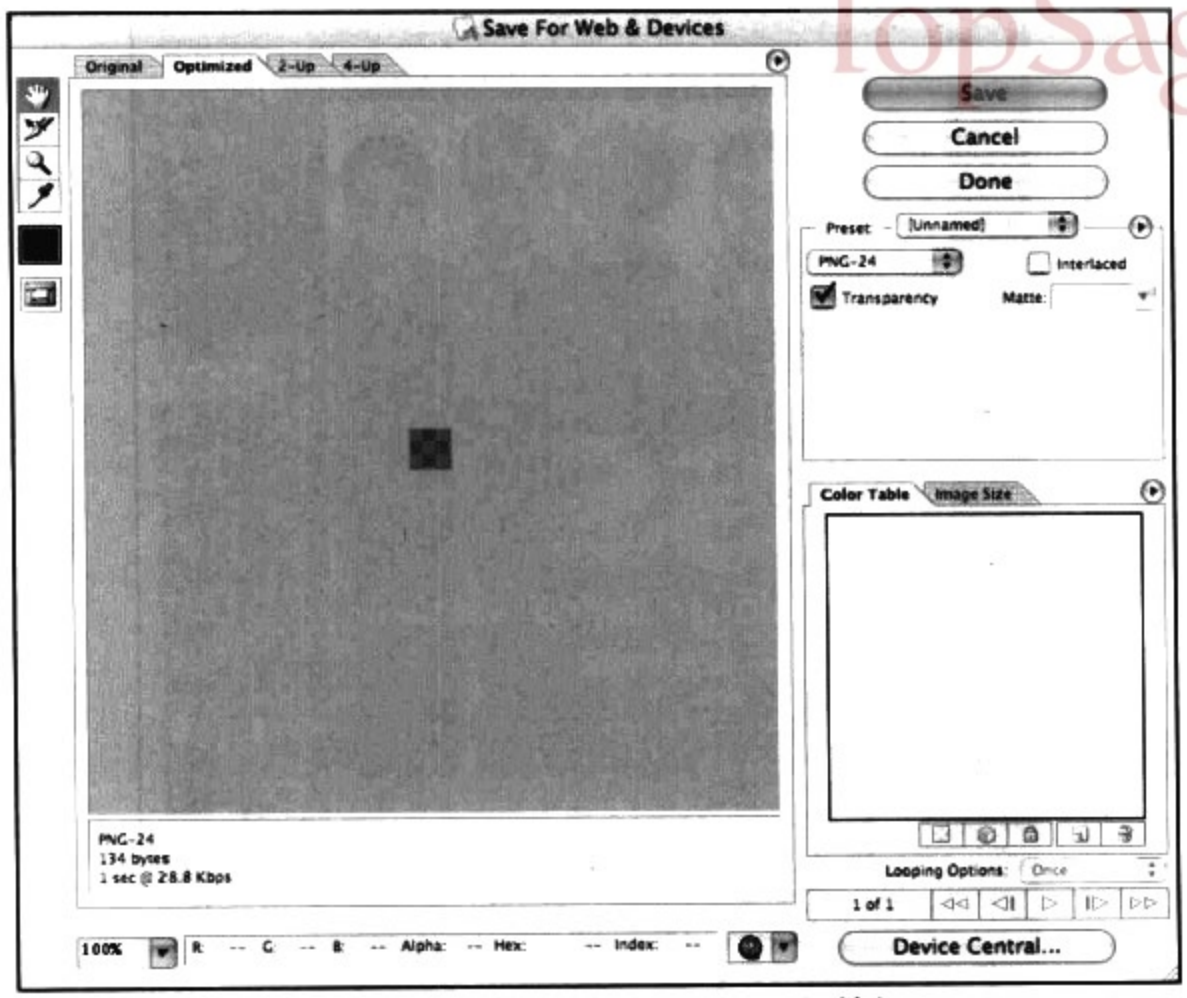


图8-9 Save for Web & Devices对话框

彩色阴影的一个有趣应用是产生一个分层效果。这要用到两个图像。第一个图像使用了彩色阴影的基本概念，但用的不是纯色，而是在垂直方向上从白色逐渐变透明的。第一个图像下面是一个复制的图像，其深红色背景更好地表现了这种渐变过程，如图8-10所示。

第二个图像是一个抽象色图像，这个图像是本书站点所用的主背景图像，如图8-11所示。

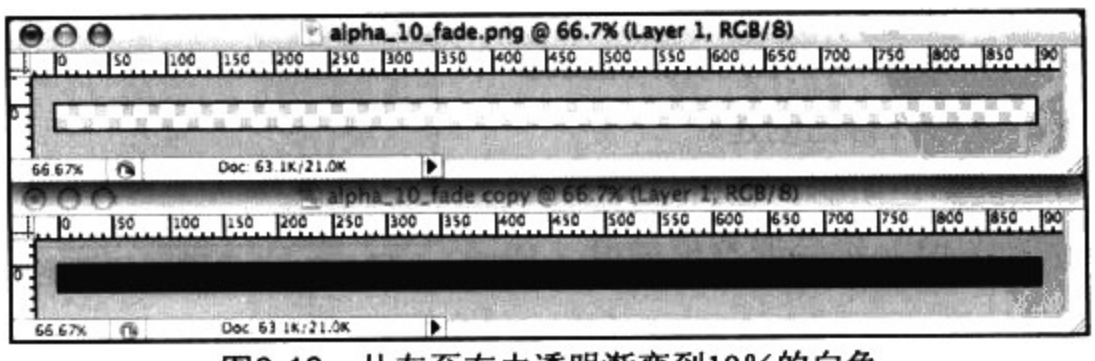


图8-10 从左至右由透明渐变到10%的白色

当背景图像之上的图层颜色发生微小变化时，这两个图像能产生很明显的效果，如图8-12所示。

甚至还能用PNG图像实现翻转效果，如图8-13所示。

```
#buybook ul:hover {
  background-image: url(/_assets/img/alpha_90_fade_blck.png);
}
```

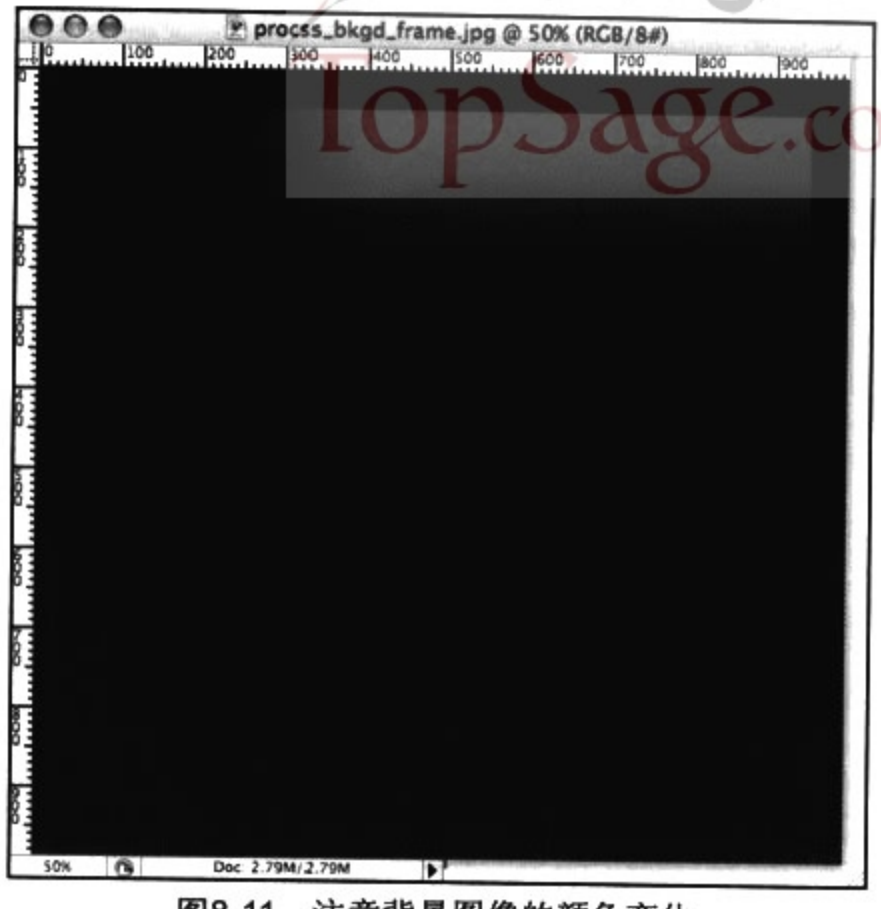


图8-11　注意背景图像的颜色变化

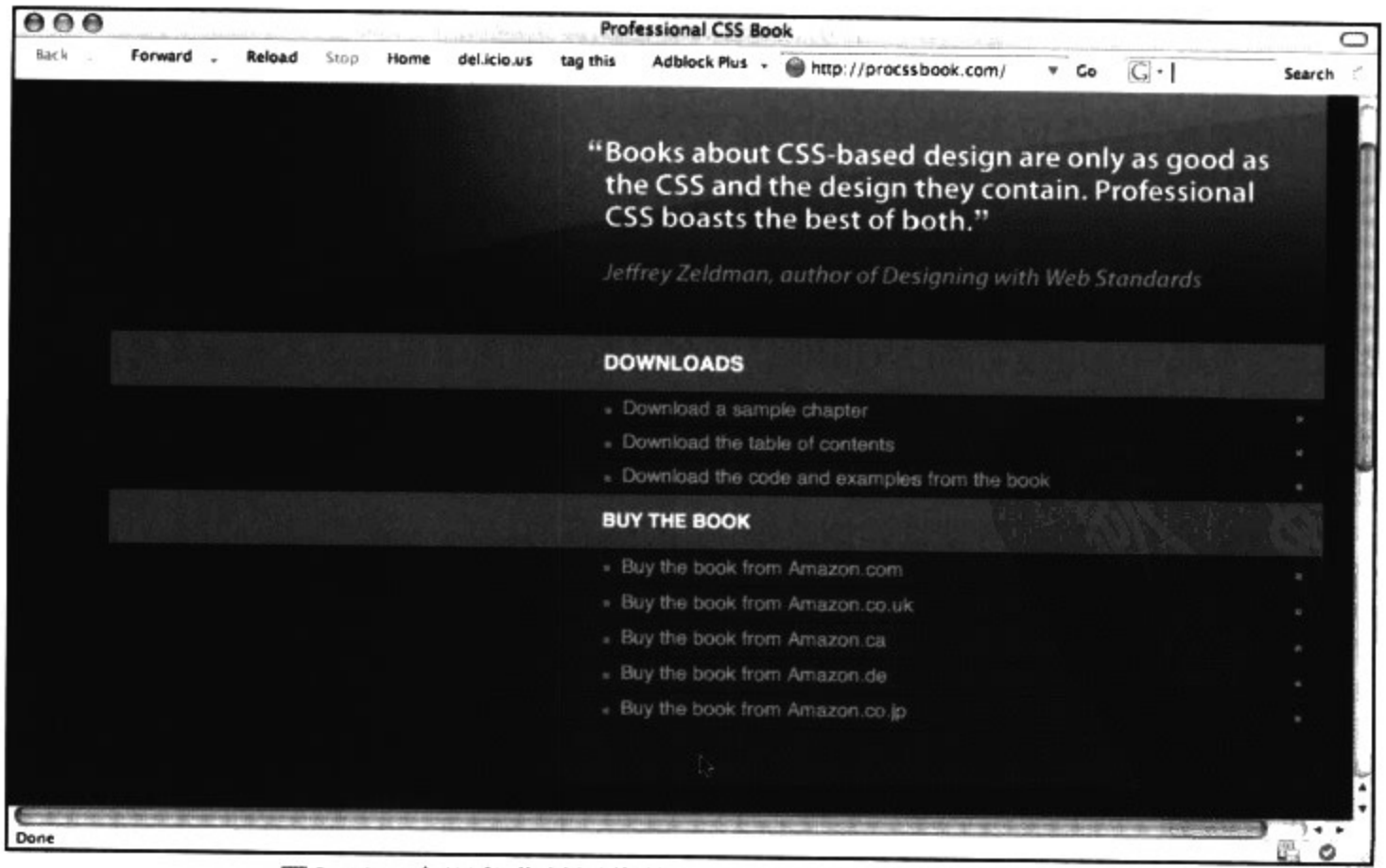


图8-12　在深色背景图像的映衬下能看到微小的颜色变化

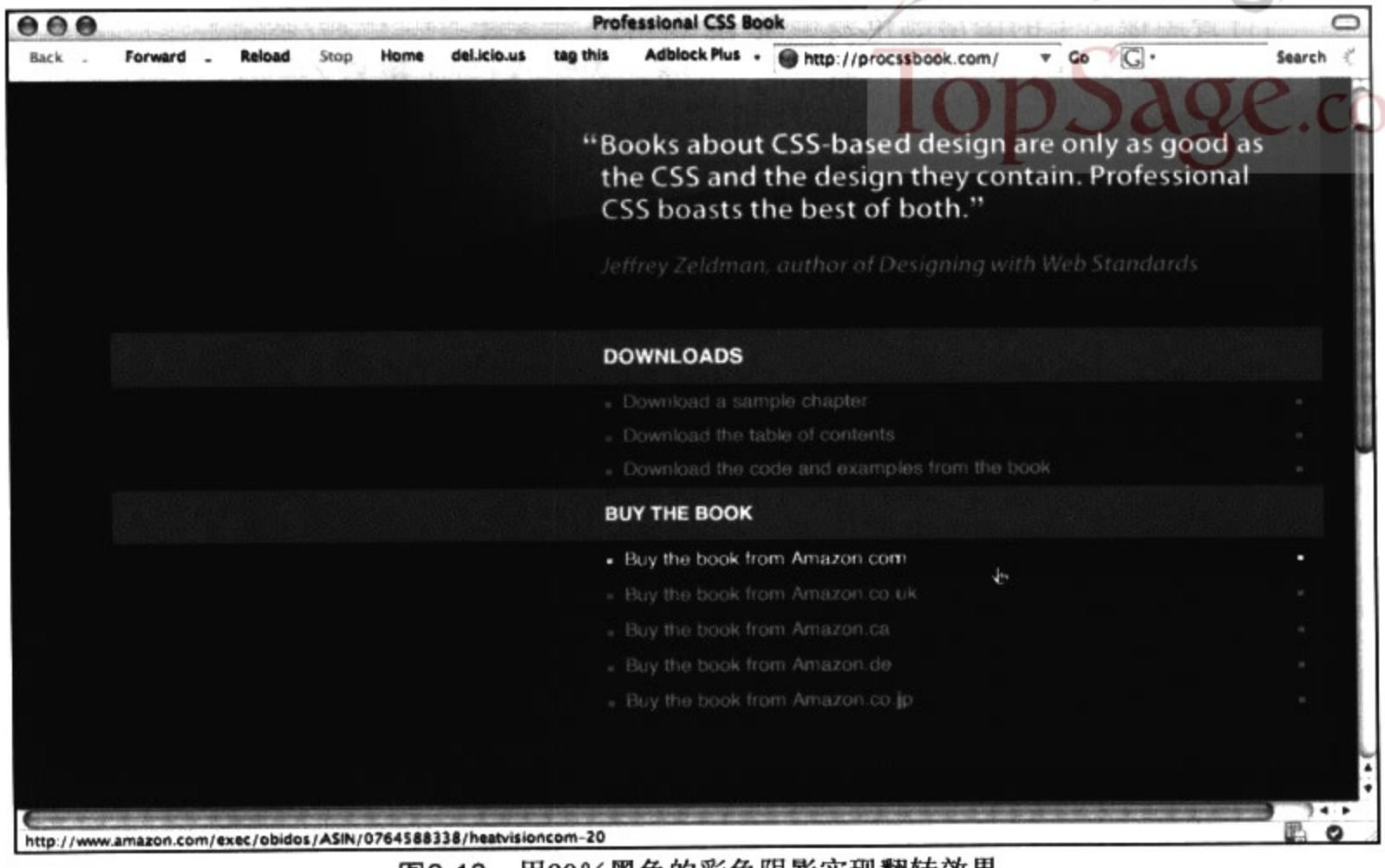


图8-13　用90%黑色的彩色阴影实现翻转效果

8.3　小结

本章介绍了透明PNG图像的用法、它的优点(和弱点)以及它们的实际应用。学习了如何在Web浏览器中使用透明的PNG图像，即使是对这种图像格式缺乏本地支持的IE 6，也存在使用透明PNG图像的变通方法。本章还介绍了颜色问题，以及如何用Alpha透明和彩色阴影使我们的工作看起来更好。

下一章将介绍如何在网站设计修改中使用CSS布局。

第9章 构建CSS布局

2003年初，一个爆炸性新闻就被越传越广。2002年10月，Wired News抛弃了传统的、以表格驱动的布局方法，而采用以CSS驱动的设计。在当时，Wired News并不是第一家进行这样重新设计的网站，但它确实最引人关注。该站点的首席设计师Douglas Bowman采用了一个可视化程度相当高的、结构非常好的商标，提交了一个非常引人注目的新设计。设计中完全采用了如CSS和XHTML这样的标准技术，正如Bowman所说，这些技术有助于“提升Web，使之远离黑暗的时代”(Wired News，*A Site for Your Eyes*: http://www.wired.com/news/culture/0,1284,55675,00.html)。

Wired News超越了学术领域和标准鼓吹热潮，正式把样式表放心地应用在主流领域。

9.1 网格与布局

布局来自网格设计的实践。通过把一个页面划分成几个重复的逻辑区域，可以从混乱状态中理出头绪来。当人们浏览该页面并希望从中找到自己需要的东西时，需要对内容的表现形式进行很好地组织。利用网格及其扩展而来的布局可以对表现形式进行很好地组织。

每天都在使用的一个有关布局的很好示例是报纸。每天报纸的头版是主题内容简介。

我们看看如图9-1(摄影theogeo, http://tingurl.com/3b3poy)所示的报纸的布局。注意这一页的文本量、不同类型的故事和信息。

图9-1　有很多栏目的报纸布局

如果不把内容放在一个多栏目布局中(如图9-2所示)，则报纸的头版看起来就很费劲

(但由于设计人员的努力，报纸在一页上能提供很多内容)。

图9-2 栏目被突出显示

当Web中的栏目用于打印时，Web也会遇到同样的困惑。例如，MidSouth Federal Credit Union对站点就进行了重新设计，老版本如图9-3所示。

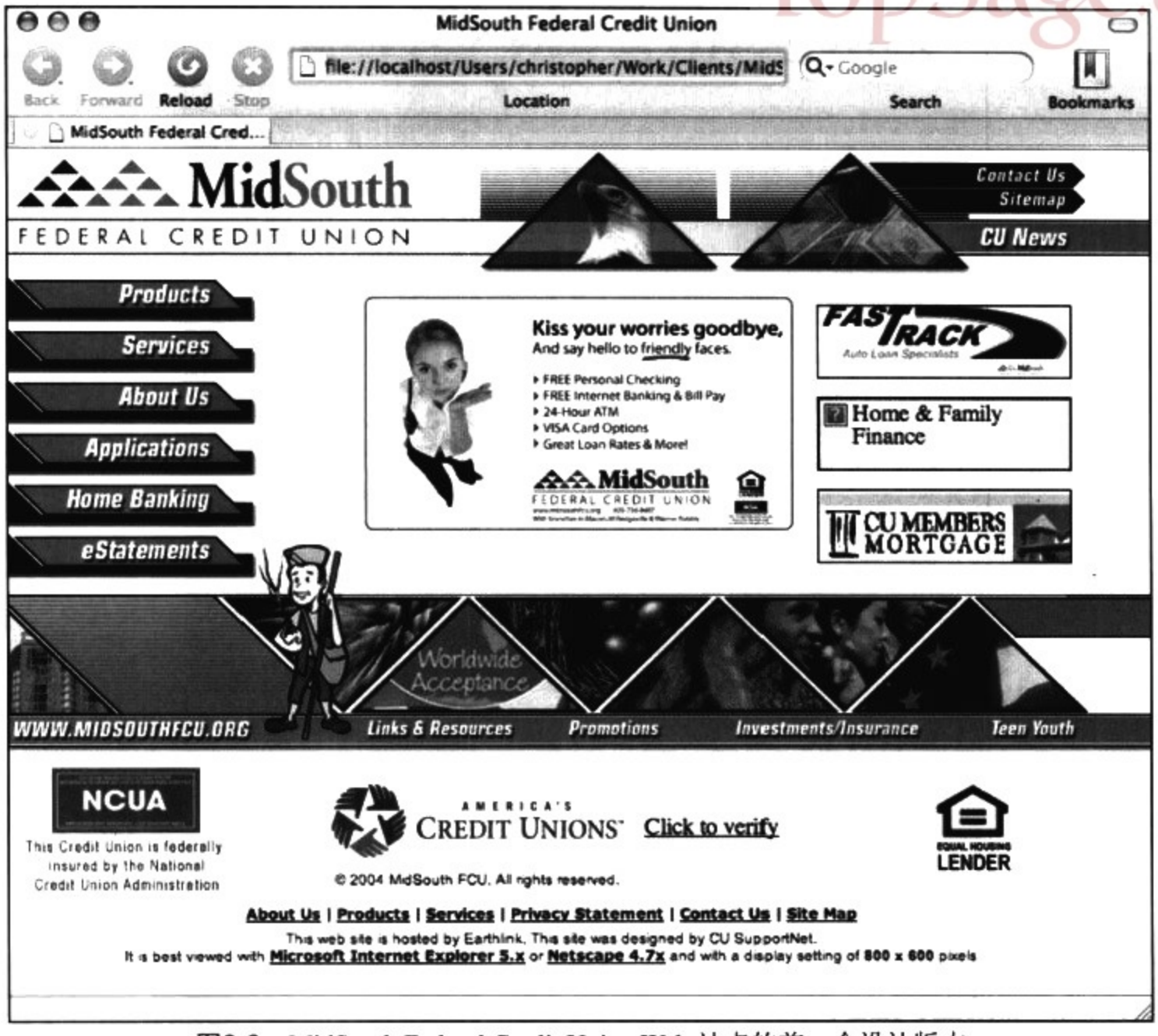


图9-3 MidSouth Federal Credit Union Web 站点的前一个设计版本

可以看出，这个主页在安排元素时避免使用空白区域。元素被放在几乎是全开放的地方。90°角或45°角被重复使用，这就导致内容在表现形式上的不一致。

在重新设计时引入了固定宽度的两栏布局，如图9-4所示。MidSouth站点在内容的表现上就变得有序了。导航菜单整齐有序，但并未被隐藏，内容之间也有分隔。

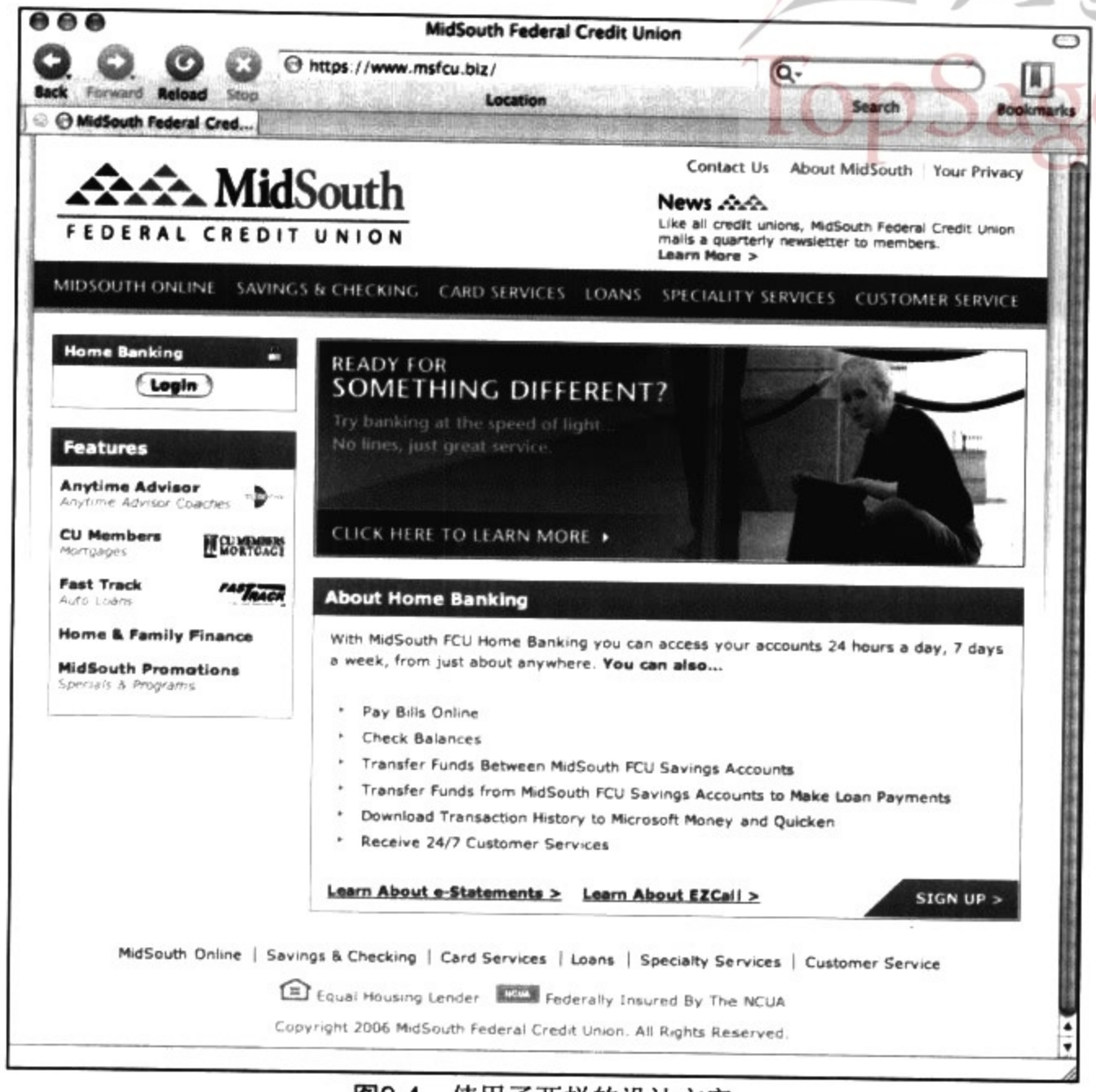


图9-4 使用了两栏的设计方案

9.2 做打印所不能做的事

打印设计所不能实现而Web媒体能实现的是布局的灵活调整。请仔细看一下Molly E. Holzschlag的Web站点(http://molly.com)，如图9-5所示。

除了鲜艳的色彩方案，Molly的站点设计是非常灵活的。Molly采用的不是固定宽度的设计，而是可以根据用户的要求调整窗口大小。图9-6是在窗口延伸到最大宽度1280×600分辨率时，中间一个主栏目的外观。

下面看看如何创建如Molly站点所用的三栏式布局。

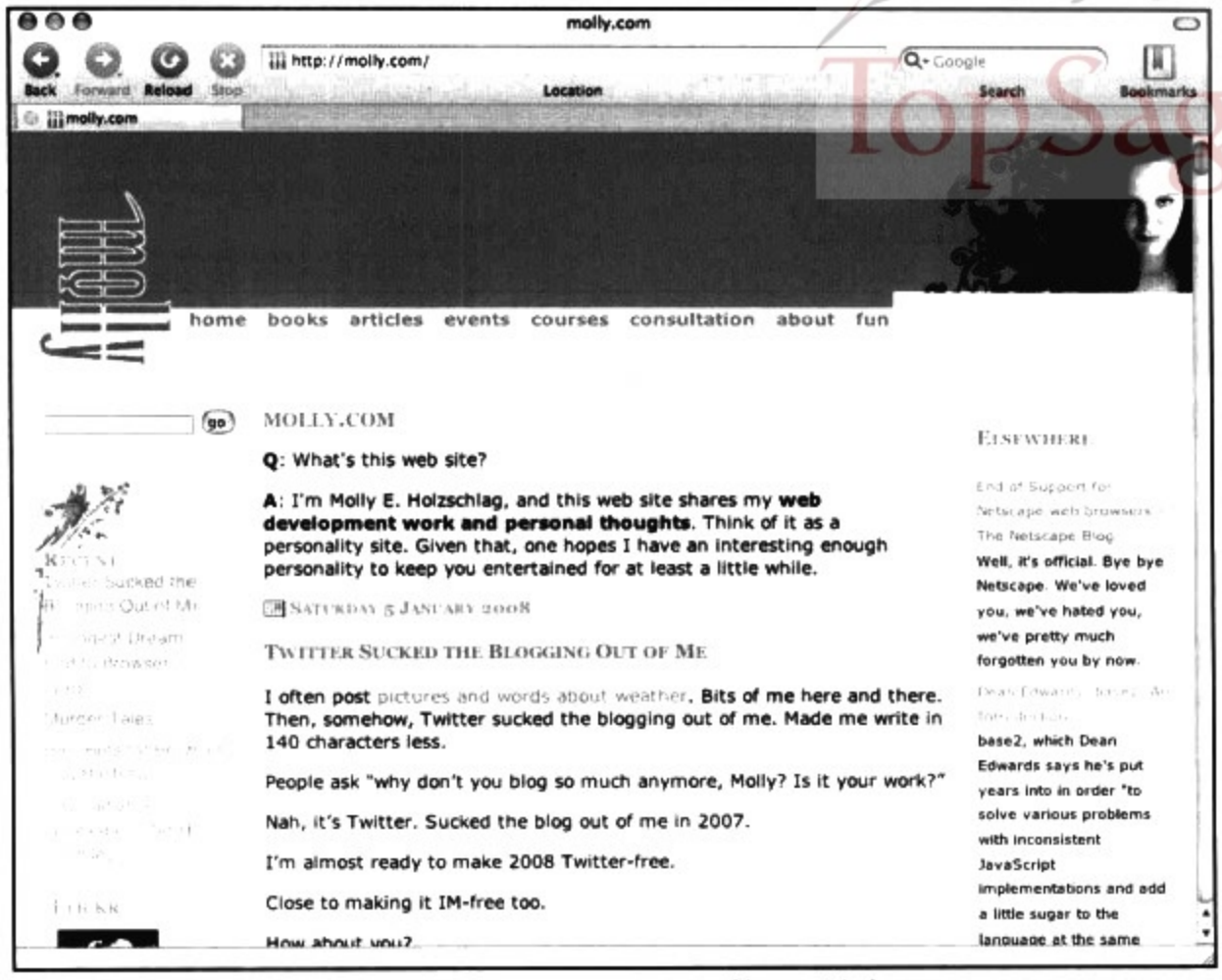


图9-5 Molly E. Holzshlag的Web站点

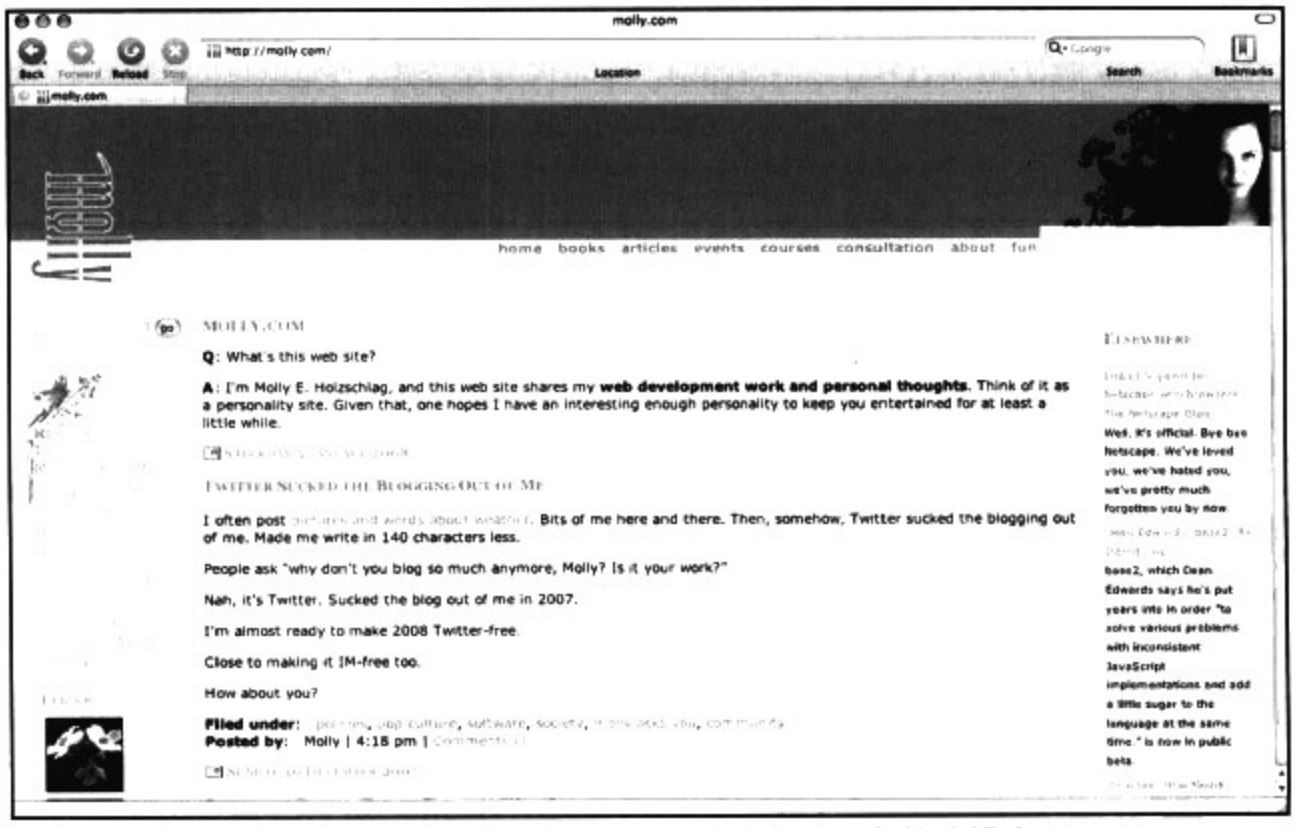


图9-6 Web站点的布局可延伸以容纳更高的分辨率

9.3 CSS定位基本原理

在复制Molly布局之前，我们先回过头来看看CSS定位的机制。在前几章已经看到，标记中的每个元素在文档流中占据一个位置。

块级元素——如题头、div、列表和段落元素——是如盒子一样一个一个堆放起来的，并在水平方向上进行扩展以占据父元素(即包含这些块级元素的元素)的整个宽度。

另一方面，内联元素是在包含它的块级元素内按垂直方向一个挨一个放置的。内联元素包括链接、图像和段元素，如em和strong。

起初，认为无样式文档内的每个元素都有一个“静态”位置，也就是说，其位置默认是不能修改的。这就是为什么段落被直接放在h2之下，而图像出现在div中的原因。

这种默认定位方案的官方术语是文档的“正常流”。如果仔细研究一个无样式的HTML文档则会发现，有边界框的元素(如p1、h1和div)都是在包含它们的块内按垂直方向放置的，是一个紧挨一个堆放的。而内联框(如span、a或img)却是在包含它们的容器内被水平放置的。由于没有任何额外的样式规则，这种默认的方案效果还不错。

显然，我们打算改变这种方案。对此我并不感到奇怪。

CSS是一种非常引人注目的布局工具，因为它能重写这些默认定位规则，且不用任何td元素就可以创建非常复杂的布局。通过编写一个简单的CSS选择符就可以把一个元素从原来的静态位置删除，您可能已猜到，这个CSS选择符为元素的position属性设置了如下所示的新值：

```
p {
  position: absolute;
}
```

除了static外，position属性还有下面三个值：

- fixed
- relative
- absolute

把position属性设置为上述三个中的任一个(非static)，就会以不同的方式从正常文档流中的位置删除该元素，而把它放在文档的另一个位置。

我们研究一下relative和absolute两个属性值和它们之间的关系，以及如何在自己的设计中更好地运用这两种定位方式。

9.3.1 功能强大的绝对定位

当采用绝对定位时，元素的位置由属性top、right、bottom和left的某些组合确定，如下例所示：

```
div#content {
  position: absolute;
 left: 10px;
 top: 100px;
}
```

在这里，left和top属性确定div(id为content)的偏移量。在标记中并没有把content div放在它之前和之后的块级元素的中间，而是把它从文档流中删除了。但它被放在什么地方了呢？

要找到答案，我们研究下列标记结构，为了简单起见，我们把该结构放在一个有效XHTML文档的body内：

```
<div id="outer">
 <p>This is a paragraph in the <cite>outer</cite> block.</p>
 <div id="inner">
  <p>This is a paragraph in the <cite>inner</cite> block.</p>
 </div>
</div>
```

这是两个相当简单的div，一个(id为inner)被嵌套在另一个(id为outer，这个名字非常清楚地说明了两者的关系)之内。每个div都包含一个子段落——这没什么疑义。显示时去掉尖括号和尖括号内的内容，只显示段落内容。下面对这段标记添加一些基本的样式：

```
#outer {
 background: #DDF;
 border: 4px solid #006;
 height: 300px;
 margin: 120px 20px 0;
}

#inner {
 background: #FDC;
 border: 4px solid #930;
}
```

设计这两个选择符的目的不是要难倒客户。第一个规则应用于id为outer的div。我们为这个div设置一个深蓝色边框和背景，并把它的高度设置为300px，然后增加外边距使之偏

离正常的位置(下移120px，水平方向上两边均偏离20px)。第二个规则给inner div设置一个浅红色背景和相匹配的边框。

是的，蓝色之上是红色。我从没说过在把这些代码示例放在一起时要考虑色彩效果。

在您由于色彩方案不合理而生气地合上本书之前，我们来看看这两个元素在页面里是什么样的。从图9-7可以看到，我在把一些基本类型信息应用于文档时也是很随便的。但随着颜色和边框被激活，就可以清楚地看到这两个div在正常文档流中是如何摆放的，inner块是outer块的子元素，目前的显示说明了这一点。

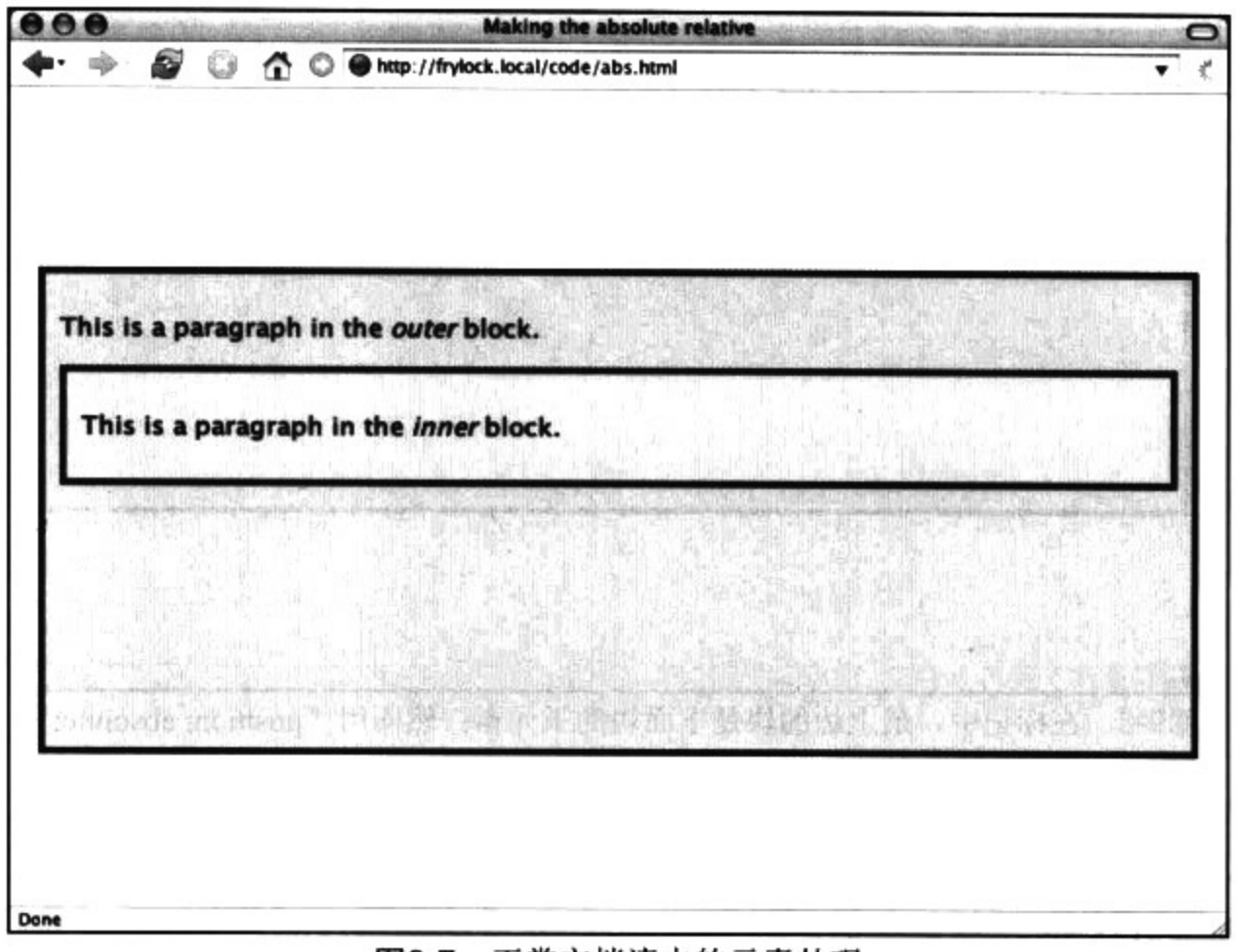


图9-7 正常文档流中的元素外观

然而用CSS修改inner元素的position属性值，就不一定能反映出设计上的这种父子关系。我们可以给#inner选择符添加简短的三行代码：

```
#inner {
 background: #FDC;
 border: 4px solid #930;
 position: absolute;
 right: 20px;
 top: 20px;
}
```

差异非常明显，如图9-8所示。在视觉上已打破了这两个div元素之间的父子关系。虽然在标记中inner块仍是outer块的子元素，但已用CSS重写了inner块在正常文档流中的位置。现在按绝对方式定位，现在的位置距文档的body元素最上边和最右边均为20px。

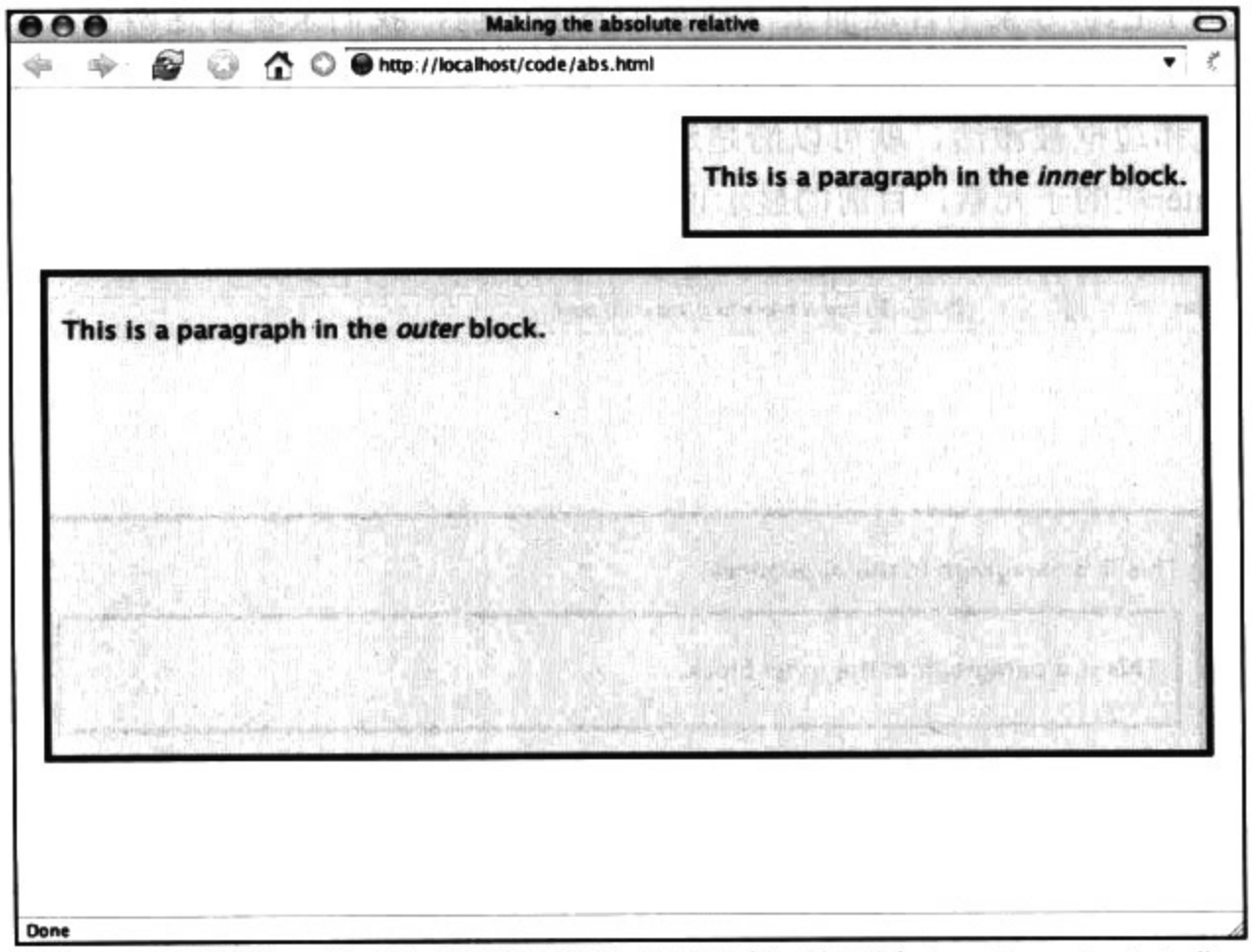


图9-8　在标记中，最上边的块是下面块的子元素；然而用“position: absolute;”把最上边的块从文档流中删除并相对于视口(Viewport)进行定位

典型情况下，这个inner块会出现在文档的正常流中，即它的上下文由它周围的其他块级元素创建。该规则重新定义了上下文，并把它放在相对于浏览器窗口边界的上下文中。这就是为什么文档的body根元素(即html元素)也被称为初始包含块的原因，因为它通常为其内的所有元素提供定位的上下文。

此外，#inner定义的新定位上下文也重新定义了子元素(即它包含的段落)的上下文。也就是说，我们不仅对inner div进行了重新定位，而且对其所包含的所有子元素也进行了重新定位。如果在这个绝对定位的块内添加一些段落就会清楚地看到这一点，如图9-9所示。

在#inner(绝对定位的块)内添加两个新的段落时，它们就继承了父块的定位上下文。这是非常好的方式，由于父块是绝对定位的，它可以继续扩展以包含新的子元素。

图9-9另一个值得注意的重要地方是，在增加了绝对定位块的高度后，外层块有些部分被遮挡了。记住，把position属性应用到一个块后，就把该块从正常流中删除了——对

于绝对定位的元素，浏览器不会为它在文档内保留任何空间。因此，绝对定位元素会覆盖页面上的其他元素，无论这些被覆盖的元素是否被定位。这是用绝对定位构建布局时要考虑的非常重要的问题，后面还将更详细地介绍这一点。

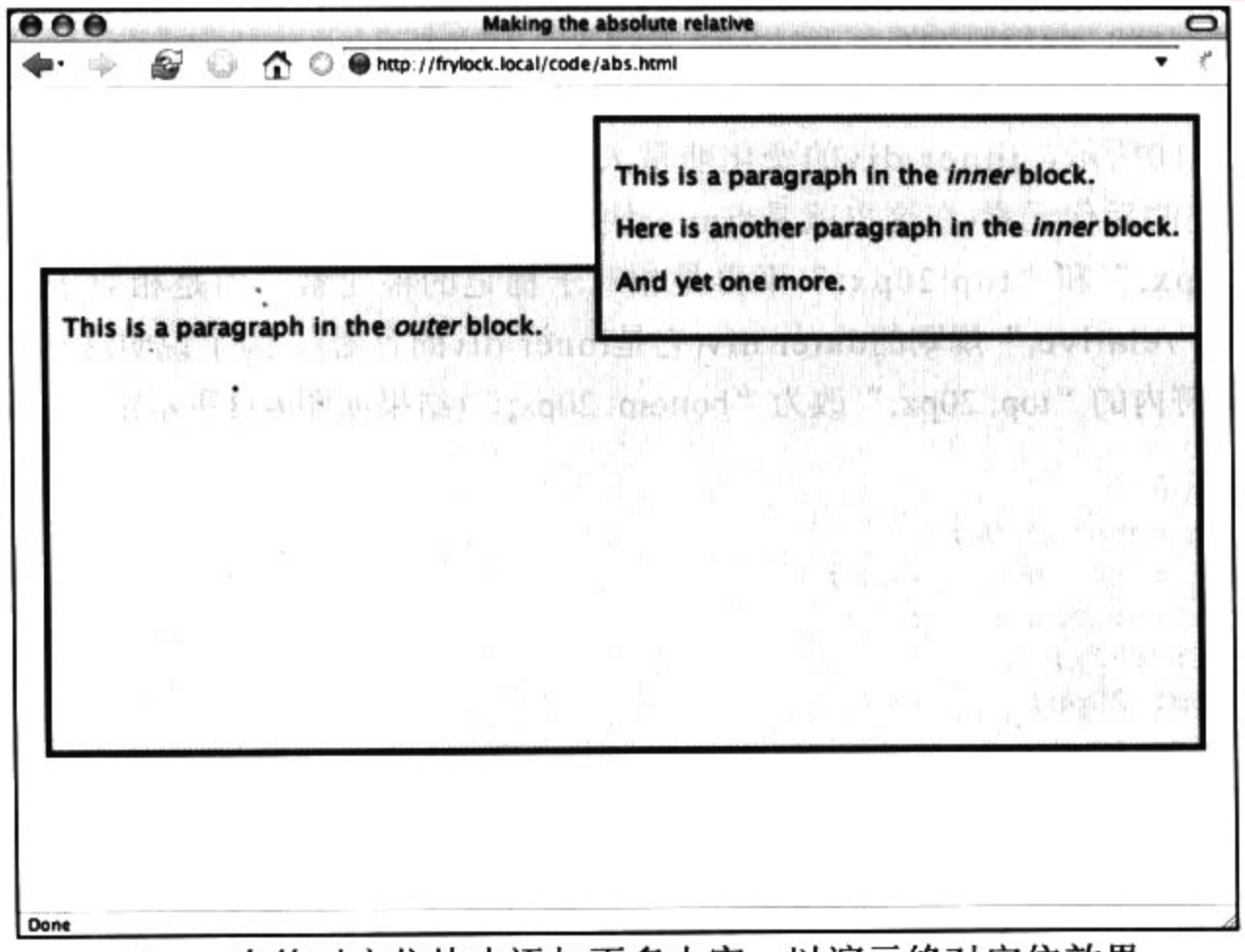


图9-9 在绝对定位块内添加更多内容，以演示绝对定位效果

9.3.2 在相对定位容器内的绝对定位

如果想对inner块的位置进行更多的控制该如何做呢？如果不希望相对于浏览器窗口进行定位，而是相对于outer div又该如何实现呢？原来我们看到的绝对定位只是默认的行为。如果不把绝对定位元素放在另一个已定位的元素——即如果它的所有祖先元素都在默认的静态位置——之内，则按图9-4所示的方式定位，即相当于最初包含块(即body元素)创建的边界进行定位。

如果您注意到了上一段中的几个“如果”，那么我很高兴不是每个人都睡着了。那么，如果绝对定位元素包含在另一个已定位元素之内，会发生什么呢？我们来看看对outer div进行一些控制后所发生的事情，这个outer div是我们在前面定义的元素，它是inner div(将要对它按绝对方式定位)的父元素。

```
#outer {
 background: #DDF;
 border: 4px solid #006;
 height: 300px;
 margin: 120px 20px 0;
 position: relative;
}
```

如图9-10所示，inner div的变化非常大。由于outer div是已定位的元素，它为所有绝对定位的后代元素(在这里就是#inner块)创建了定位上下文。因此inner div的偏移“right:20px;”和“top:20px;”不再是相对于标记的根元素，而是相对于我们应用了“position: relative;”规则的outer div(它是inner div的容器)。为了说明这一点，我们把#inner选择符内的“top: 20px;”改为“bottom: 20px;”(结果如图9-11所示)：

```
#inner {
 background: #FDC;
 border: 4px solid #930;
 position: absolute;
 right: 20px;
 bottom: 20px;
}
```

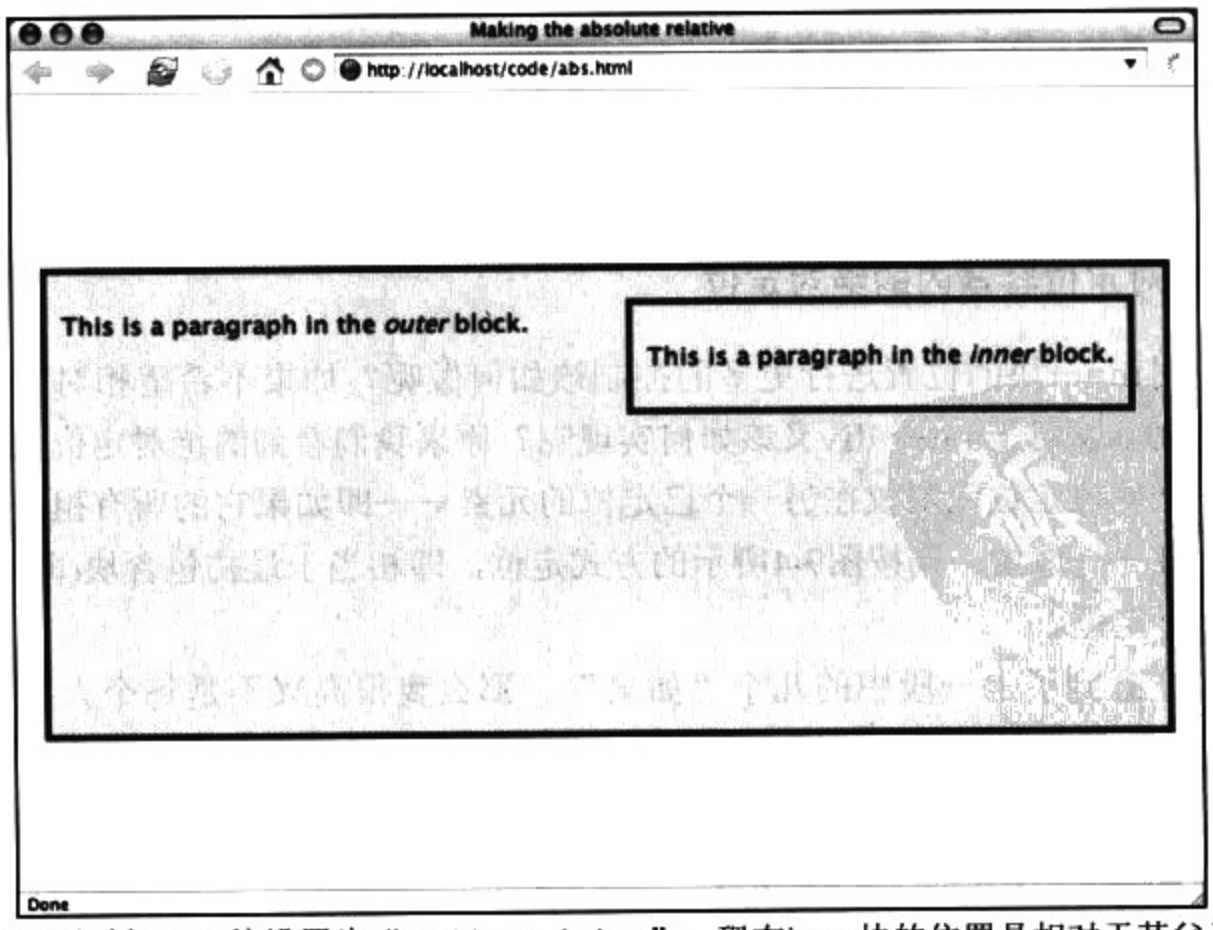


图9-10　通过把outer块设置为“position: relative;”，现在inner块的位置是相对于其父元素的

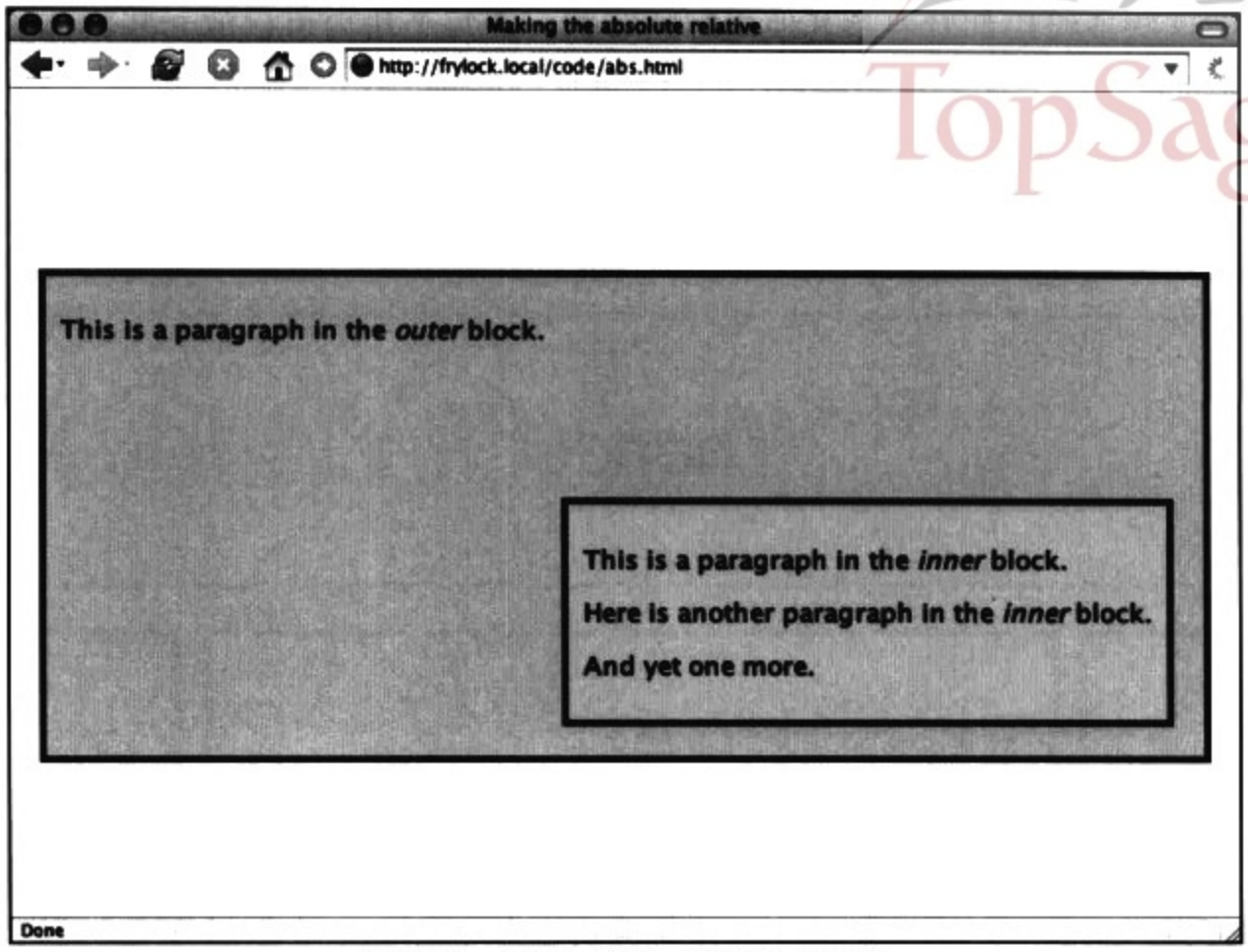


图9-11　CSS的作用：靠底边的定位

在对inner定位时，不是在inner上边界和inner父元素的上边界之间创建一个20px偏移量，而是在inner下边界和inner父元素的下边界之间创建一个20px的偏移量——所要做的只是在选择符内添加一行而已。通过对本章的继续学习，我们将更详细地介绍这种方案的好处。从现在起，相对定位容器内的绝对定位元素之间的关系将成为在Molly.com样式中创建灵活的、三栏布局的基础。

9.4　创建三栏：布局的基础

正如在把哈佛大学主页转换为全CSS/XHTML的布局(参见第1章)时，创建的三栏布局必须是小巧的、有良好意义的标记。要实现这一点，我们先快速浏览一下该页的内容(如图9-12所示)。

本章主要介绍这个布局的下列几个方面：

- 横跨页面top栏的水平标题。
- 中间栏，包含重要内容和其他感兴趣的事件。
- 左栏和右栏，包含一些辅助内容，如子导航、广告等。

我们把这些顶级块都看着是其他内容的容器。在每个块内可以放一些具体的内容，应用CSS规则对每一部分的表现形式进行精确控制。

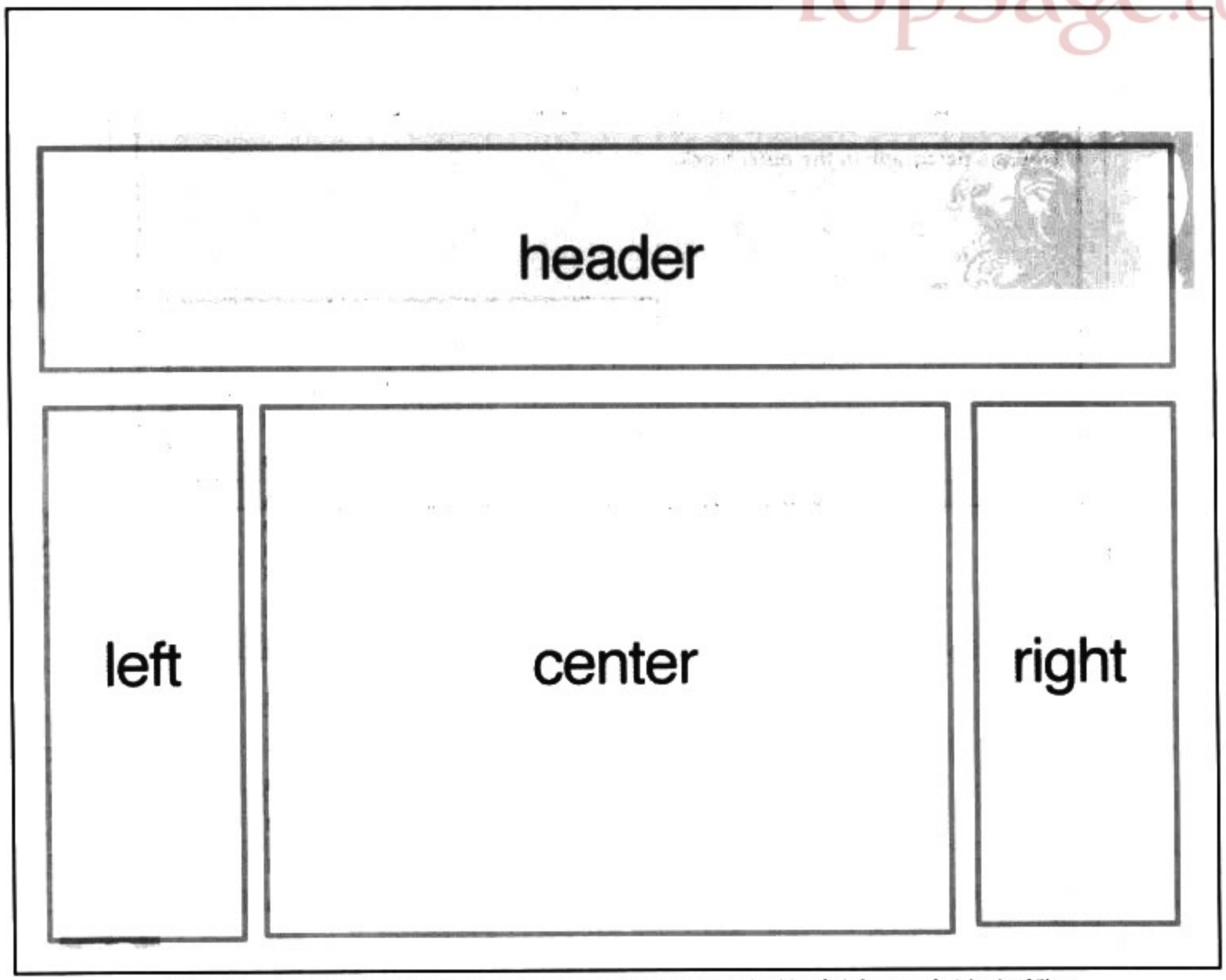


图9-12　标识需要在标记中体现的内容区(由于页面长度原因没有显示页脚)

建立这个灵活的三栏布局是本章的主要目标。一旦建立了这个布局，就可以对设计进行更精细的控制，直到满意为止。因此我们把精力放在布局框架的建立——标题栏、内容栏、左栏和右栏——使这个页面看起来是真正专业的设计。

9.4.1　编写XHTML：从模式到标记

记住我们的目标：创建一个基本的标记文档来反映这个框架，如程序清单9-1所示。

程序清单9-1：三栏布局的标记基础

```
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>My 3-column layout</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  </head>
  <body>
  <div id="header">
    <p>How do you like them apples?</p>
    <hr />
  </div>
  <div id="left">
    <h2>This is the left column.</h2>
    <p>Some basic information goes here.</p>
    <!-- More content here -->
  </div>
  <div id="right">
    <h2>This is the right column.</h2>
    <ul>
      <li>Did you know that lists rock?</li>
      <li>They do.</li>
      <li>Quite a bit.</li>
    </ul>
    <!-- More content here -->
  </div>
  <div id="content">
    <h1>Welcome to my page layout.</h1>
    <p>Certe, inquam, pertinax non ero tibique, si mihi probabis ea...</p>
    <!-- More content here -->
  </div>
  <div id="footer">
   <hr />
   <p>Them apples were <em>tasty</em>.</p>
</div>
</body>
</html>
```

三个内容区的映像如图9-12所示，我们仅在页面上对这四部分进行了标记——即用div元素进行了标记。每个div有一个唯一的id，反过来，id又代表相应的文档区：第一个div是header、接下来是left，等等。我们还在页面底部包含了一个footer块。

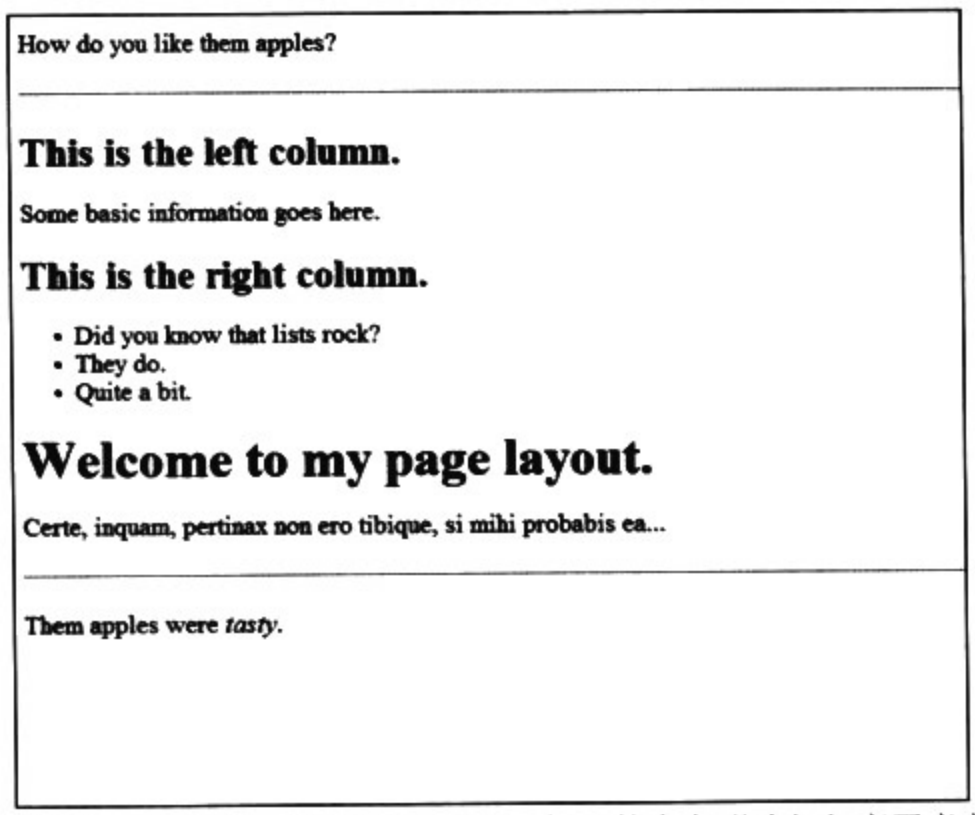
说明：

在样本标记(和本章其他地方)中所用的id是为了演示的目的。根据元素的外观命名，或根据屏幕的位置命名，就可以把标记与特定的表现形式有效地结合在一起。在重新设计

网站时，如果#left div突然显示在页面的上边或右边会是什么样子？

我们还要考虑元素内所包含内容的意义，据此对这些元素进行命名。或许#right比#advertising更好，#left比#subnav更好。这里没有正确答案，我们应尽量使名字的描述性更强一些，确保样式与结构有更明确的分离。

在页面的每一部分都可以放一些简单的内容作为占位符。然而，即使是对一些无用信息编码，也应该有一定意义：页面上最重要的标题用h1标记，左、右栏的标题用h2标记，页面文本的其他内容用正确的列表和段落元素标记。如果没有任何样式规则，标记看起来确实相当平淡，如图9-13所示。


How do you like them apples?

This is the left column.

Some basic information goes here.

This is the right column.

- Did you know that lists rock?
- They do.
- Quite a bit.

Welcome to my page layout.

Certe, inquam, pertinax non ero tibique, si mihi probabis ea...

Them apples were *tasty*.

图9-13　无任何样式的HTML文档，标记的意义明确但却毫无意义

在这一阶段显然需要意义明确、语义丰富的标记。在用样式表控制站点的表现时，重要的是要考虑如何让不能识别CSS的浏览器用户访问站点内容。在呈现像图9-9所示的页面时，用户可能看不到您细心规划的设计。然而，由于在标题和页脚div中使用了水平线(hr元素)，人们仍能访问站点的内容。在使用CSS隐藏了设计中的这些元素之后，在不能识别CSS的老浏览器上，这些元素对内容有很好的分隔作用。

对更多的细节，可采用在标记中所用的两级标题元素。徜徉在sans CSS的无视障用户很快就能看出Welcome to my page layout比栏标题更重要。屏幕阅读器可读出标记题头所用的级别，使视力有障碍的用户能很快了解到页面的层次。可访问性倡导者马上会指出，Web上最盲的用户是Google——它采用的是很陈旧的方式，不关心表现形式，只关心内

容。使用这种智能标记不仅能帮助人类用户对页面进行导航，还能帮助搜索引擎更好地了解如何分配权重和进行检索。

9.4.2 样式层

稍微偏离点主题，我们对内容应用一些基本的表现规则，如程序清单9-2所示。

程序清单9-2：把一个基本的CSS规则应用到文档

```
body {
 color: #000;
 font: 76%/1.5em "Lucida Grande", Verdana, Geneva, Helvetica, sans-
serif;
 margin: 0;
 padding: 0;
}

h1, h2 {
 font-family: "Trebuchet MS", Verdana, Geneva, Helvetica, sans-serif;
 font-weight: normal;
 line-height: lem;
 margin-top: 0;
}

#header {
 background-color: #DFD;
 border: 2px solid #060;
}

#footer {
 background-color: #FDD;
 border: 1px solid #C00;
}

#left, #right {
 background-color: #DDF;
 border: 2px solid #00C;
}

#header hr, #footer hr {
 display: none;
}
```

第一个规则把一些基本的颜色和字体信息应用到body元素，这些信息被其文档树中的后代元素继承，更确切地说，是在第二个规则重写它之前。通过把不同的font-family值应

用到文档内的标题元素(h1和h2)，我们就可以对继承来的样式进行重写，从而采用不同的样式而不是默认样式。我们希望它们在视觉上有更突出的效果。

下面三个规则把边框和背景色应用到文档的主区域：标题为亮绿色、页脚块为显眼的红色、左右栏为醒目的蓝色。在这里，应用这些规则只是为了对比，不是为了华丽——从图9-14可以看到具体的效果。

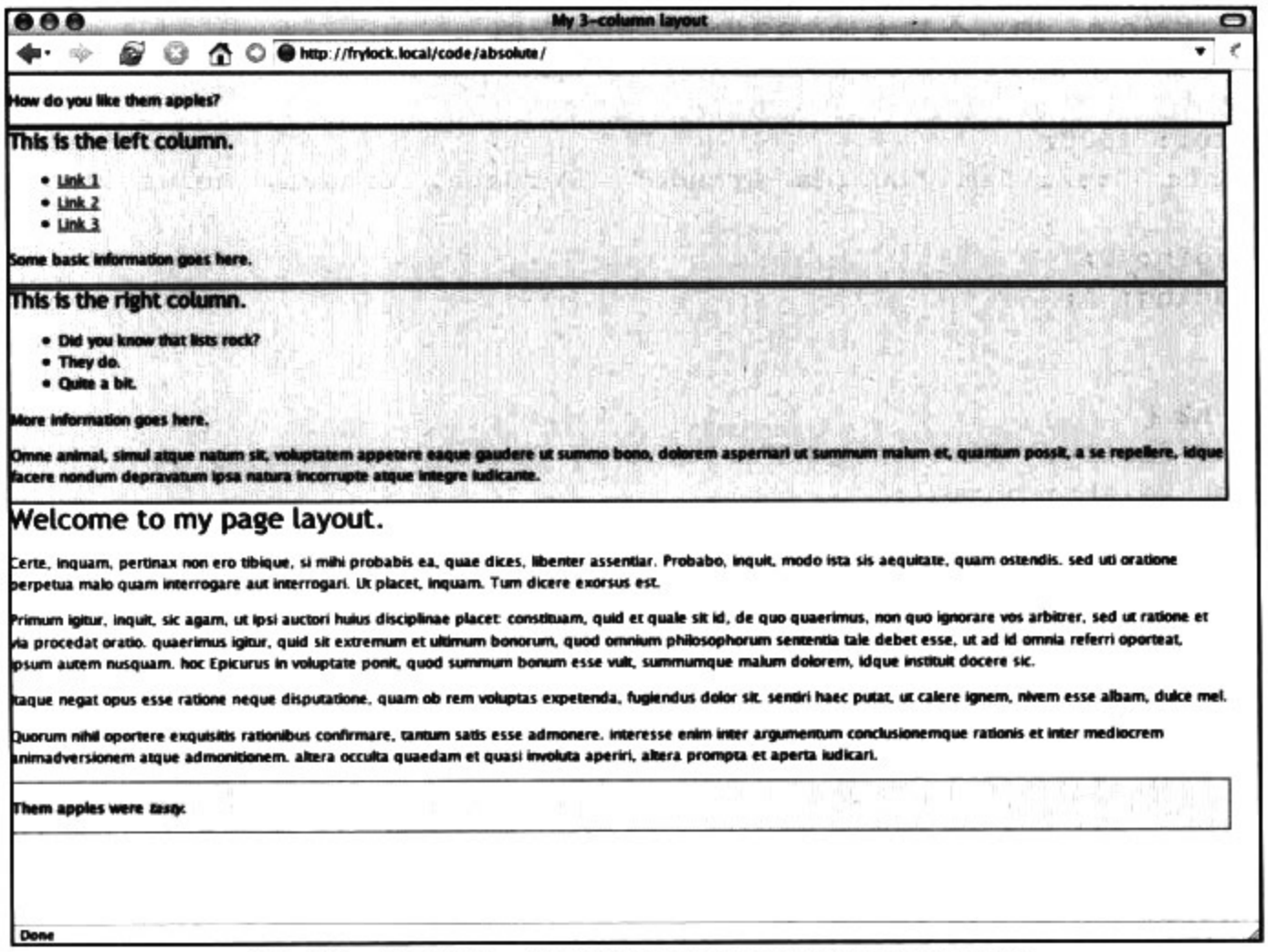


图9-14　用CSS把一些字体和颜色信息应用到HTML文档

1. 用偏移量引入定位规则

对不同内容元素的边界有更好的了解后，我们现在对布局进行充实(如图9-15所示)：

```
#left {
 position: absolute;
 left: 0;
 top: 0;
 width: 175px;
}

#right {
```

```
    position: absolute;
    right: 0;
    top: 0;
    width: 175px;
}
```

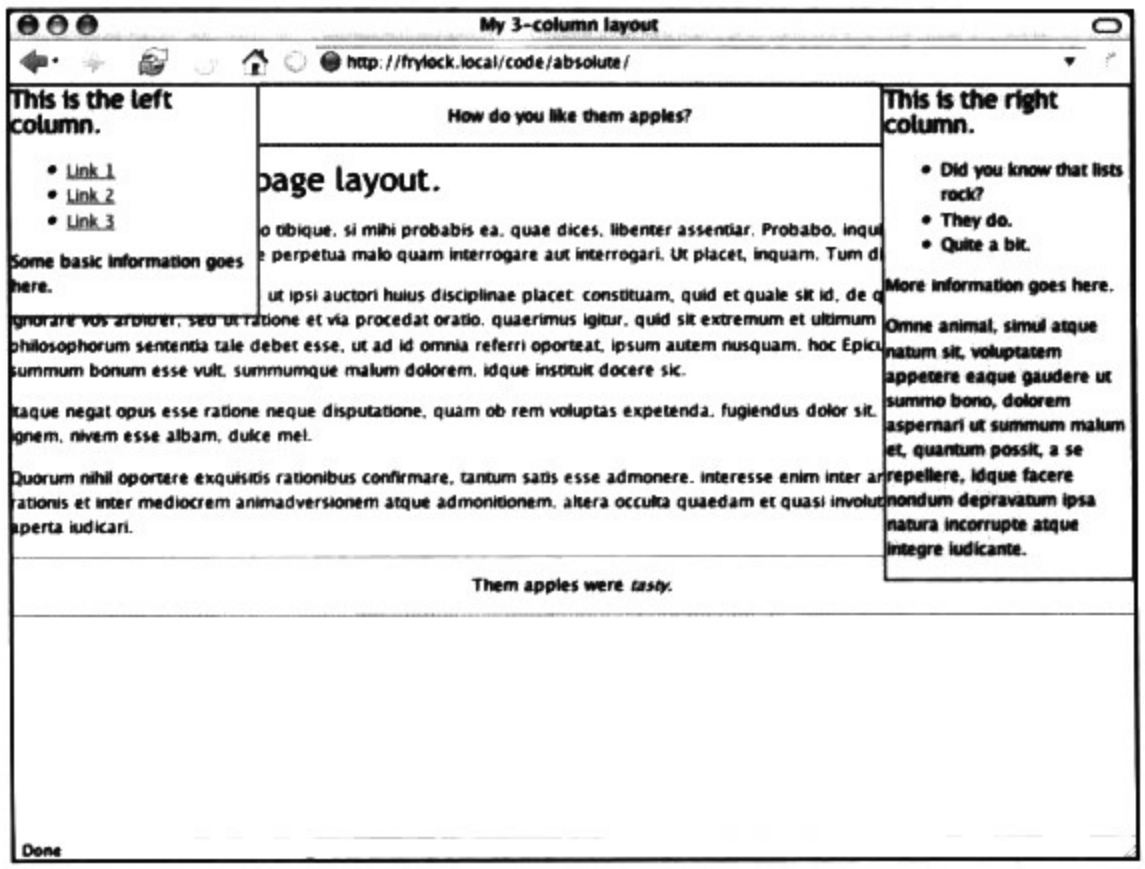


图9-15 按照绝对定位把左右栏放在了预定位置，但还没达到预期目的

这两个选择符把左、右块从正常文档流中删除并按绝对方式定位。为每个div都显式定义了宽度175px，然后放置在窗口最顶端(top：0)。指定的水平偏移量(对于#left为“left:0;”，可以猜到对于#right则为“right:0;”)对左、右栏进行进一步的控制，并创建了三栏布局的起点。“左”、“右”块开始与名称相符了，栏也开始成形了。

但可以看出布局还远未完成。左、右栏与标题栏、页脚栏和内容div重叠了。通过对这两个块进行绝对定位，可从正常文档流中把它们完全删除。这意味着，在我们能按像素对它们进行精确定位的同时，文档流中的其他元素不需要再为它们保留空间了。为了对布局进一步完善，还需要做一些额外的工作。

2. 无条件保持相对性

这两个块相对于html定位的唯一原因是它们不是一个已定位元素的后代。在您开始思索这个问题时，记住这一点很有帮助。如果改变这一点会发生什么呢？

在程序清单9-3中，我们把这三个无标题的块封装在id为container的div中。

程序清单9-3：在标记中添加一个容器

```
<div id="header">
  <p>How do you like them apples?</p>
  <hr />
</div>
<div id="container">
  <div id="left">
    <h2>This is the left column.</h2>
    <!-- More content here -->
  </div>
  <div id="right">
    <h2>This is the right column.</h2>
    <!-- More content here -->
  </div>
  <div id="content">
    <h1>Welcome to my page layout.</h1>
    <!-- More content here -->
  </div>
  </div>
    <div id="footer">
  <hr />
  <p>Them apples were <em>tasty</em>.</p>
</div>
```

我们承认，这个容器对文档的整个语义没有多大价值——标记纯化论者可能用“表现hack(presentational hack)”这个术语来表达，这仅是为了达到一些设计目标而添加的元素。但正是因为有了这个容器div，才可以对它应用一个三行的CSS选择符，它将把一些正确度量值保存到站点布局中(如图9-16所示)：

```
#container {
  position: relative;
}
```

现在可以把这个容器看成是一个已定位的元素，因此所有从这个容器派生的定位元素都具有了一个新的上下文，这个上下文就是由这个容器创建的——您已猜到，这些派生元素就是左、右栏目块。左、上、右偏移量不再是相对于html元素，而是相对于容器div。这意味着，当这个容器元素向下扩展和水平增长时，左、右块本身会重新进行定位。也就是说，如果在标题栏的纵向尺寸增加(如图9-17所示)时，容器div新的水平位置就会在它绝对定位的子元素中反映出来。

虽然标题栏再一次可见，但页面内大多数内容仍是不可读的。内容和页脚仍可能被绝对定位的兄弟遮挡住了。如何才能使内容区更灵活呢？

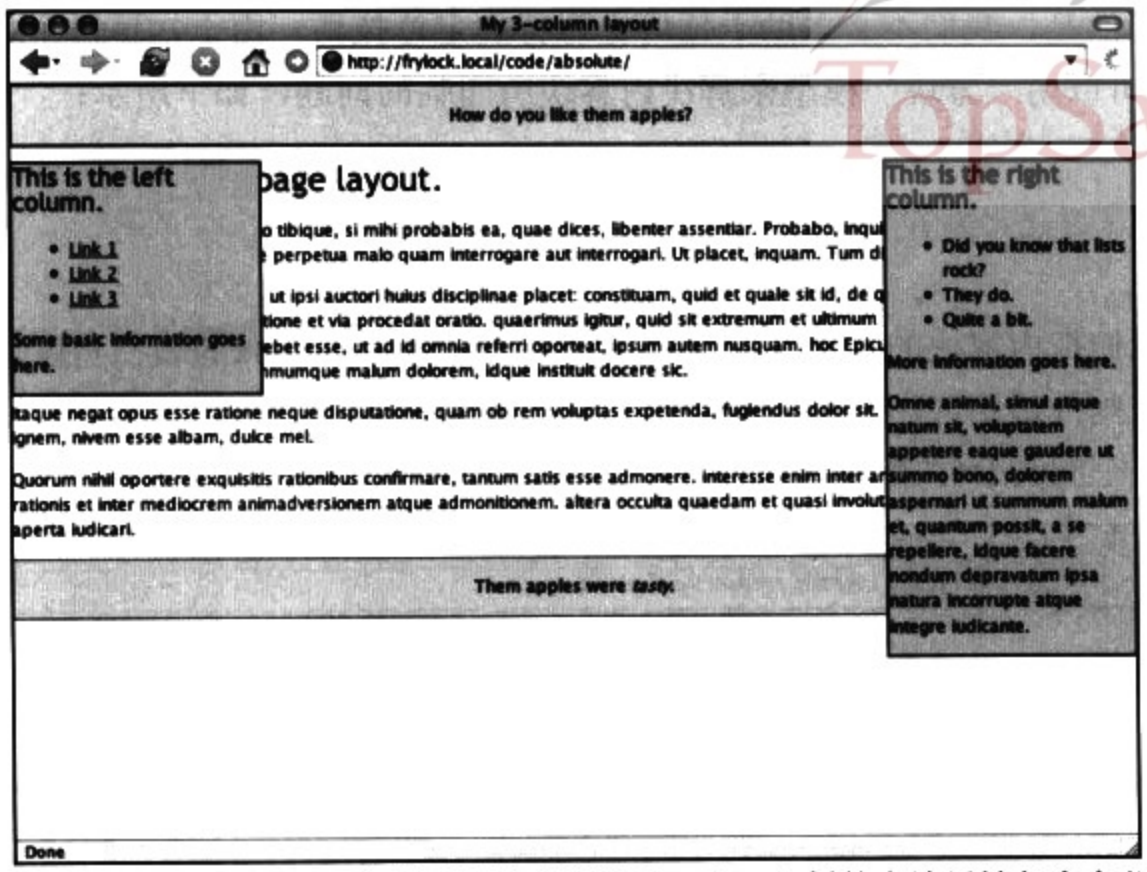


图9-16 把“position: relative;”应用到新的容器块后，左、右栏的上边界被包含在父容器内；但页面内容呢?

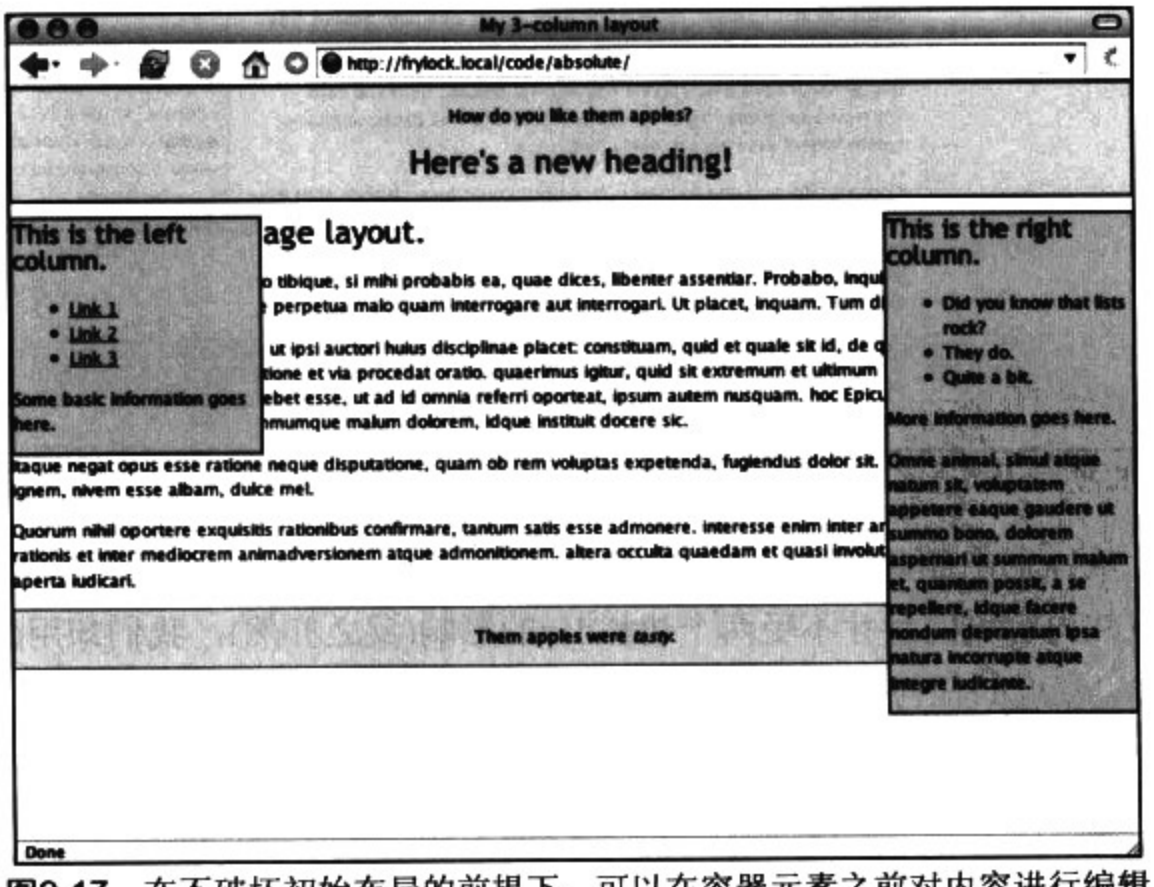


图9-17 在不破坏初始布局的前提下，可以在容器元素之前对内容进行编辑

值得庆幸的是，我们不必求助于任何更奇特的定位技巧。由于知道每个边栏的宽度固定为175px，可以用“方块”模型来解决内容块所面临的问题，如下所示：

```
#content {
  margin: 0 190px;
}
```

在这里我们把内容块的两个水平外边距设置为190px：175px的边栏宽度加15px的空白区域。当把这些外边距应用到这个内部块的左、右两边时，它的计算宽度被压缩并与两边栏之间的可视空间吻合(如图9-18所示)。

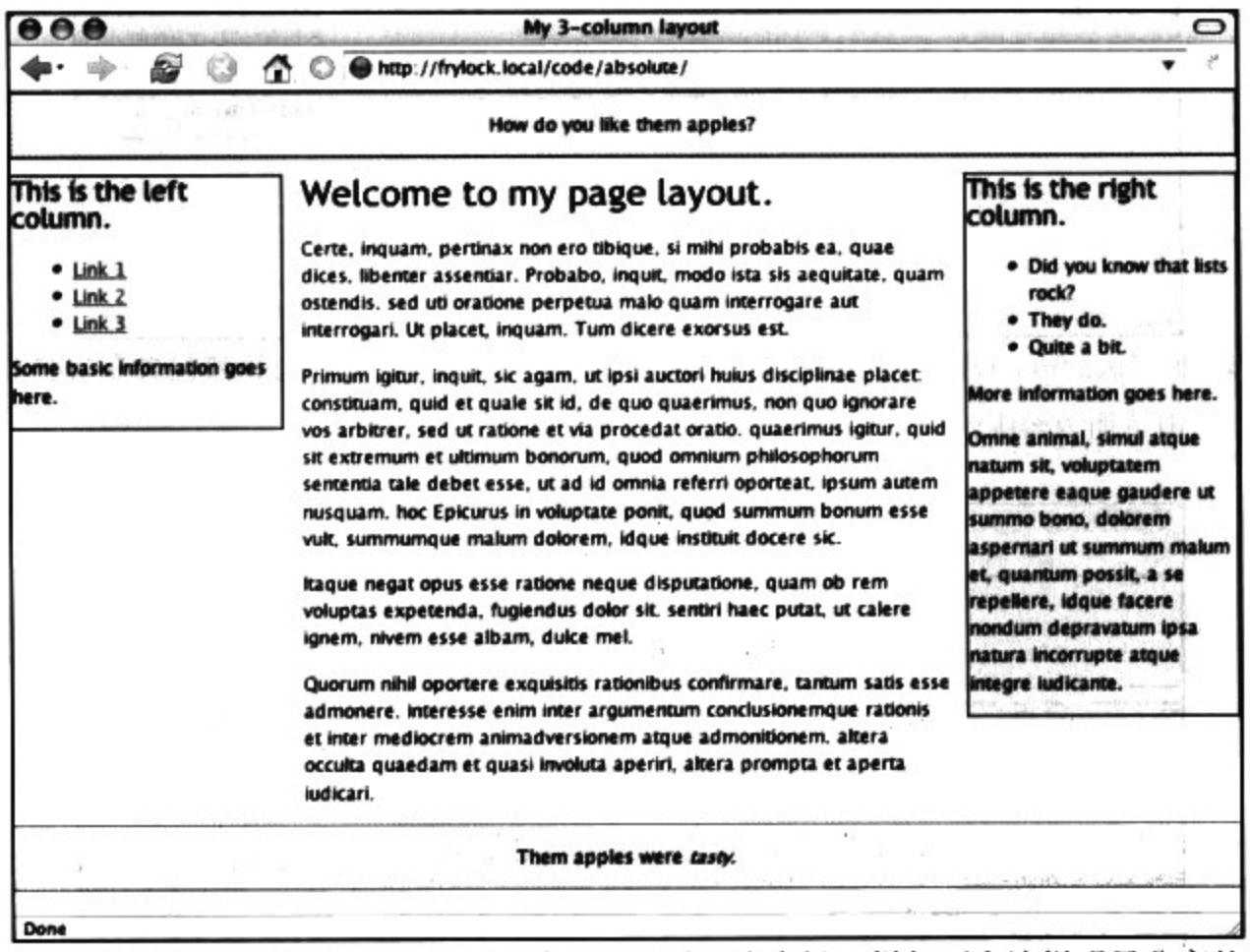


图9-18 根据两边栏之间的宽度添加内边距，可以形成中间一栏与两个边栏明显分离的幻觉

如果暂时从标记中删除两个边栏(或用CSS隐藏)，就可以更好地看到这种效果(如图9-19所示)。虽然内容块的尺寸不受两个边栏div的影响(反之亦然)，我们却用CSS实现了这样的效果。窗口宽度的任何变化都会导致整个页面内容的重新调整。

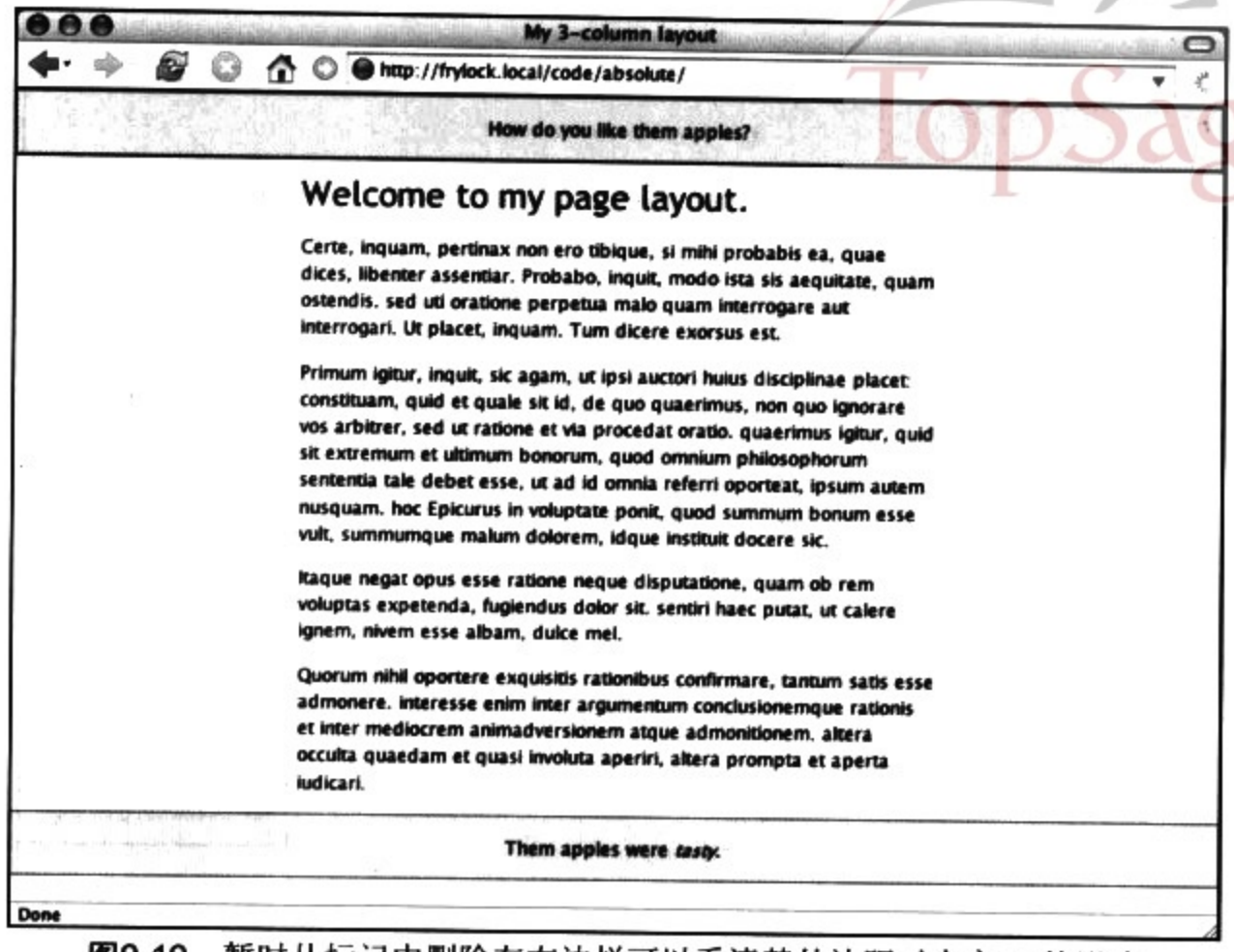


图9-19 暂时从标记中删除左右边栏可以看清楚外边距对内容div的影响

遗憾的是几乎不能得到如图9-19的效果。当我们在页面布局的顶边和水平定位获得令人满意的效果时，页脚会出现一些问题。当两个边栏元素相对于容器div定位时，它们不能相对于容器div进行大小调整。如果边栏高度超过了容器高度，在设计中位于它们之下的元素就很容易出现问题，图9-20就是这样的示例。在右边栏中添加一些额外的段落，很快就把页脚挡住了。

前面已看到，相对于绝对定位元素而言，正常文档流中的元素(如标题块、内容块和页脚块)遵守不同的定位上下文。这就是最初看到的标题块和内容块出现重叠的原因，现在页脚块又出现了相同的问题。令人高兴的是，我们可以采用针对内容块的相同方法。了解了“方块”模型后，我们可定义页脚div周围的外边距，又一次实现了这种中间“栏”效果，它独立于它两边的边栏。

因此，按照同样的方式，我们把页脚div在水平方向上的两个外边距定义为190px。对于布局的其余部分，有两种选择：

- 写一个新的CSS选择符，该选择符定义页脚元素的外边距。
- 在标记中，把页脚div移到内容块中。这样就可以把页脚有效地包含在自己计算的宽度中。

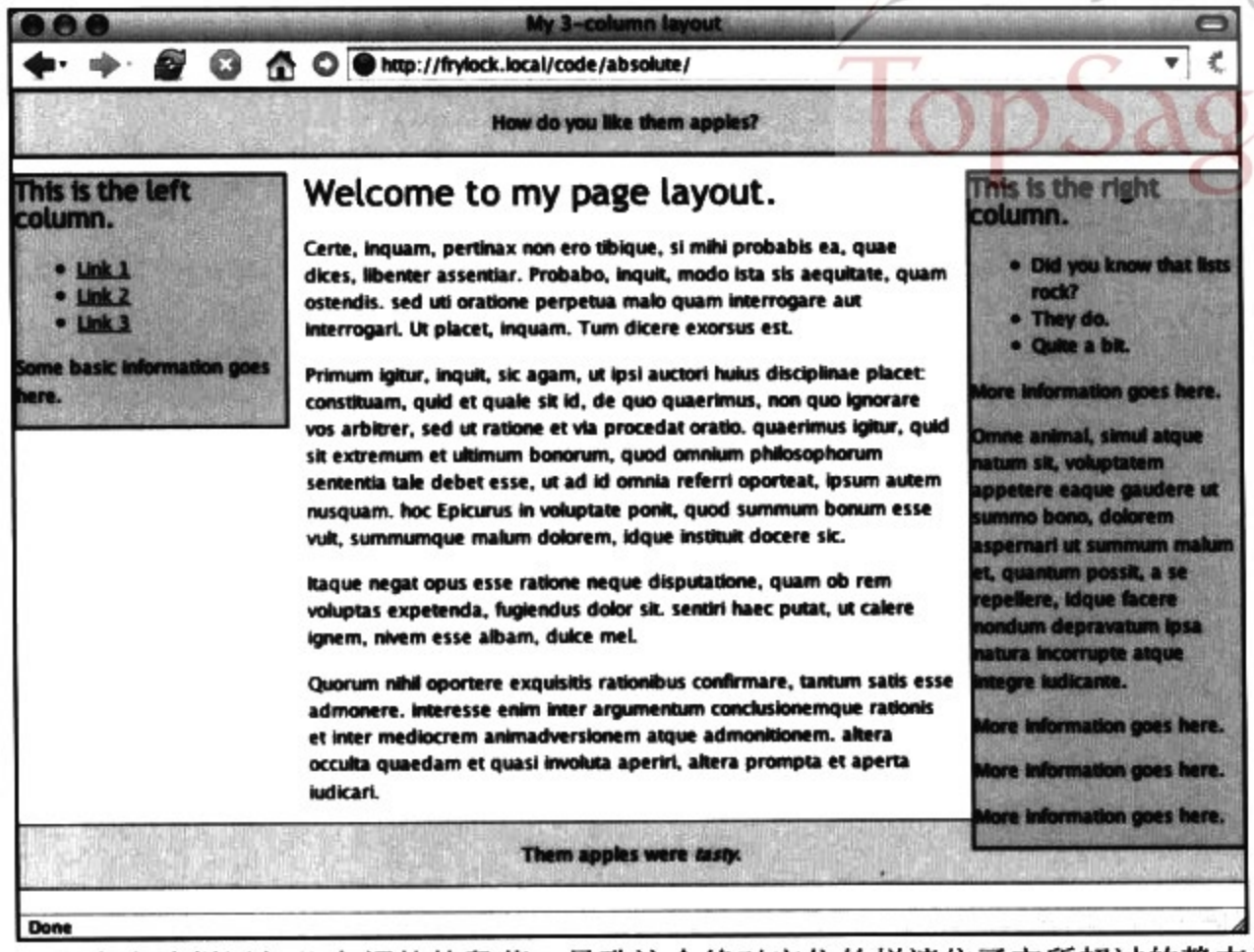


图9-20　在右边栏添加几个额外的段落，导致这个绝对定位的栏遮住了它所超过的静态元素

两个解决方案的效果相同。无论是给页脚div本身定义外边距，还是给其容器定义外边距，页脚总是出现在两个边栏中间，如图9-21所示。

为了防止绝对定位边栏的任何遮挡，可以把页脚div放在主内容块内。然而，当边栏高度超过了中间一栏的高度时，页脚就明显高于页面底部。

这是绝对定位模型的一个非常严重的缺陷。绝对定位元素可以从文档流中删除，被放在一个离奇的位置，确实是这样。但作为设计工具，绝对定位最致命的是不知道每个已定位元素周围的元素内容。绝对定位元素不仅会遮住非定位元素，也可能遮住其他应用了“position: absolute;”的块。这就是CSS设计人员主要用悬浮模型(float model)来控制布局的原因。

说明：

事实上，有非CSS的方案可以解决这种绝对定位模型的“底部遮挡(bottom blindness)”问题。Shaun Inman是一个知名的Web设计师和开发人员，编写了一些非常棒的JavaScript(http://www.shauninman.com/mentary/past/absolutely_positive.php)以自动清除绝对定位引起的遮挡。

当然，在把任何一个迂回的解决方案(CSS、标记或其他)应用到站点前应该进行充分测试。它们在解决了所遇到问题的同时可能带来支持和维护方面的其他问题。

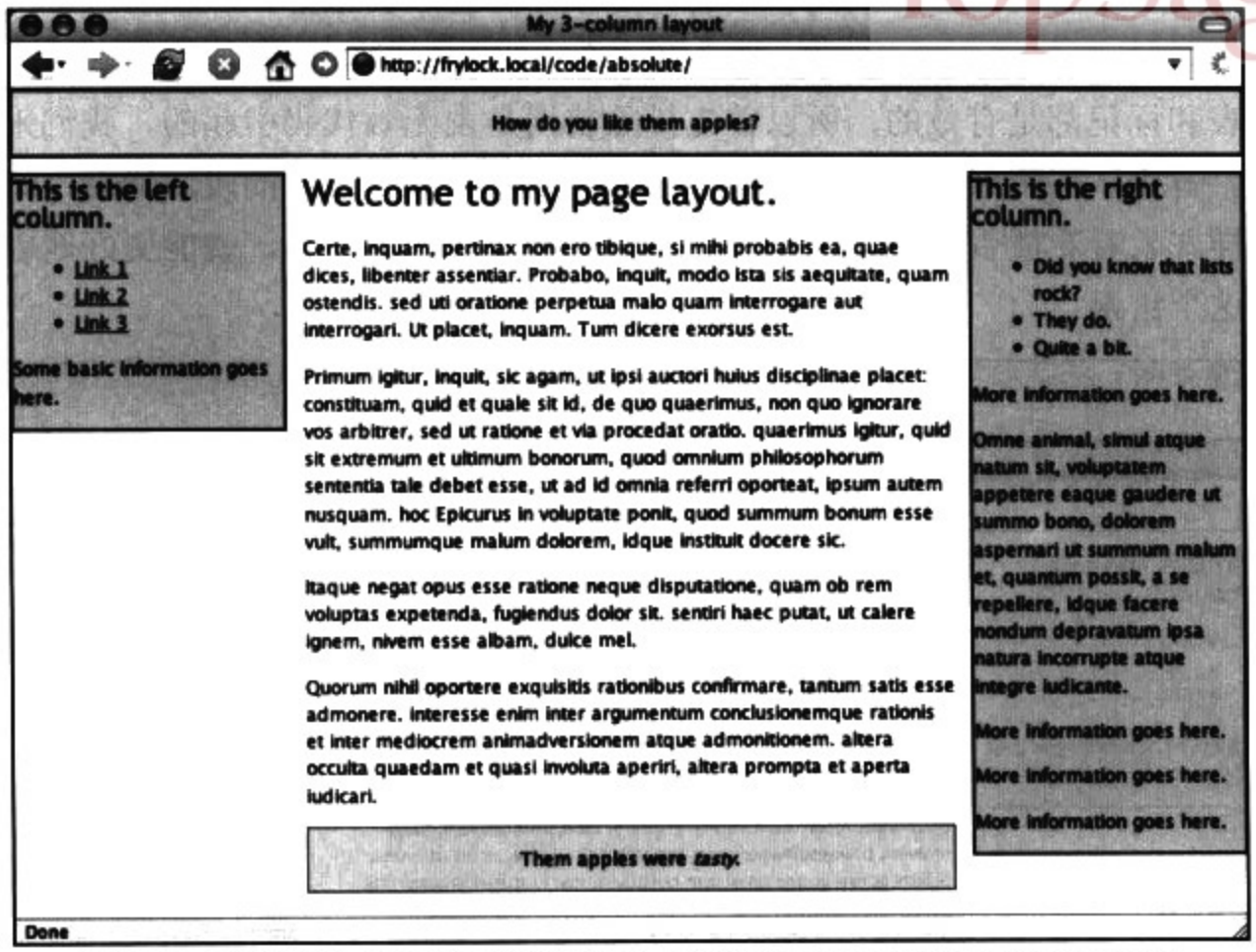


图9-21　给页脚div定义同样的外边距，解决了被绝对定位元素遮挡的问题

9.4.3　解决浏览器错误

您的布局在所选浏览器内看起来很不错，标记有效、样式表的精准程度极高。因此您的设计自然在人类所知的所有浏览器中都非常完美，对吗？真是这样吗？

如果您对此深信不疑，这就像您驾驶一叶行驶缓慢的小舟准备到大海中一样不可思议。虽然当今的浏览器对层叠样式表有丰富的支持，但支持程度却有非常大的差异——正如我常说的，所有浏览器都是不同的。不幸的是，有效代码跨浏览器平台后并不能得到同样完美的显示。由于每个浏览器都有一些错误，因此必须对代码在各个平台上进行充分测试。而且常常还必须引入一些与浏览器相关的hack，以保证设计能以预期的结果显示给所有用户。

我们来看看设计中遇到的两个错误，并介绍一些迂回的解决方案。

1. IE 5/Mac

在IE 5/Mac中打开三栏布局时，所有一切似乎都显示得很好——没有注意到浏览器窗口的最下面的样子(如图9-22所示)。要修复这个小小的问题，首先是要找到问题之所在。由于样式表和标记都是有效的，所以首先排除错误是由无效代码引起的。我们采取排查错误的方法。首先，编辑部分标记看看能否把问题限制在一个特定的段。然后，编辑一个针对该段的样式表看看能否解决这个问题。一旦找到了错误的原因，就能更好地创建一个补丁来修复这个错误。

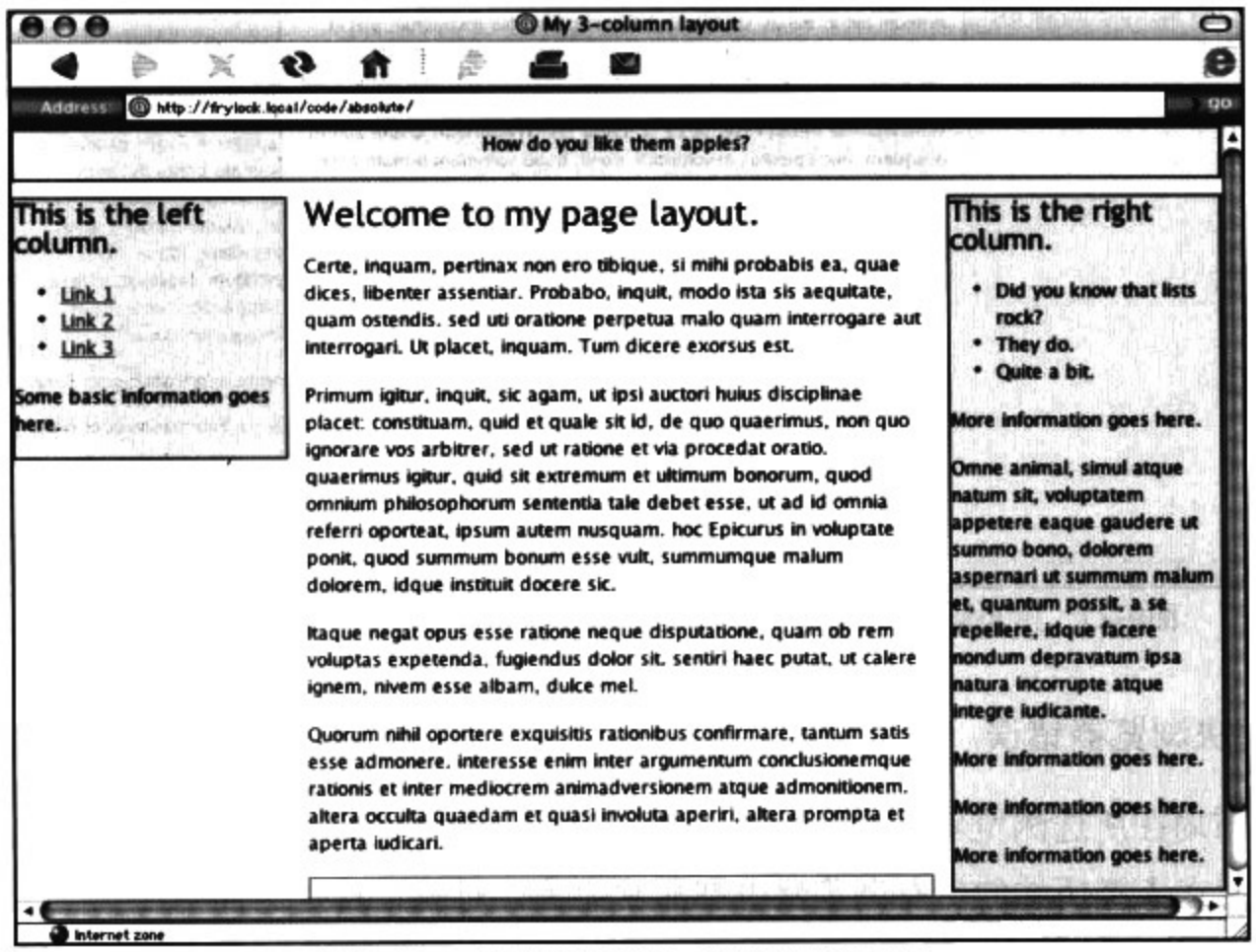


图9-22　水平滚动条是如何出现在这里的

说明：

附录D中有一些解决与浏览器有关问题的妙招。

牢记这个过程，看看能否从标记中把错误找出来。由于问题出在页面的右边栏，或许这是查找问题的最佳起点。暂时从标记中删除整个right div，然后重新加载这个页面，我们的怀疑得到证实：水平滚动条消失了！在删除最右边一栏后，发现问题的核心是滚动条

问题(如图9-23所示)。虽然现在我们对错误定了位，但如何确定引起问题的真正原因呢？更重要的是如何修复呢？

接下来就是一些繁琐的编码、删除、Web搜索和测试等工作。由于每个浏览器都有一些自己的特性，首次遇到的问题总是令人头疼的。一旦遇到更多的问题并积累了解决问题的丰富经验，调试过程就不那么费时了，也不再令人头疼了。在浏览器具备对Web标准更统一的支持之前，对任何一个设计项目，这种修复问题的测试阶段都是很费时的。我猜测您可能要等到问题完全被确定后才往下进行，但永远乐观的人是不会等的。

经过一些试验后取得了一些小小的突破。修改right属性的值使滚动条消失了。特别地，任何大于或等于15px的值都可以修复这个问题，小于15px就会出现问题。但是这个解决方法不太理想，因为布局看起来不是太完美(如图9-24所示)。

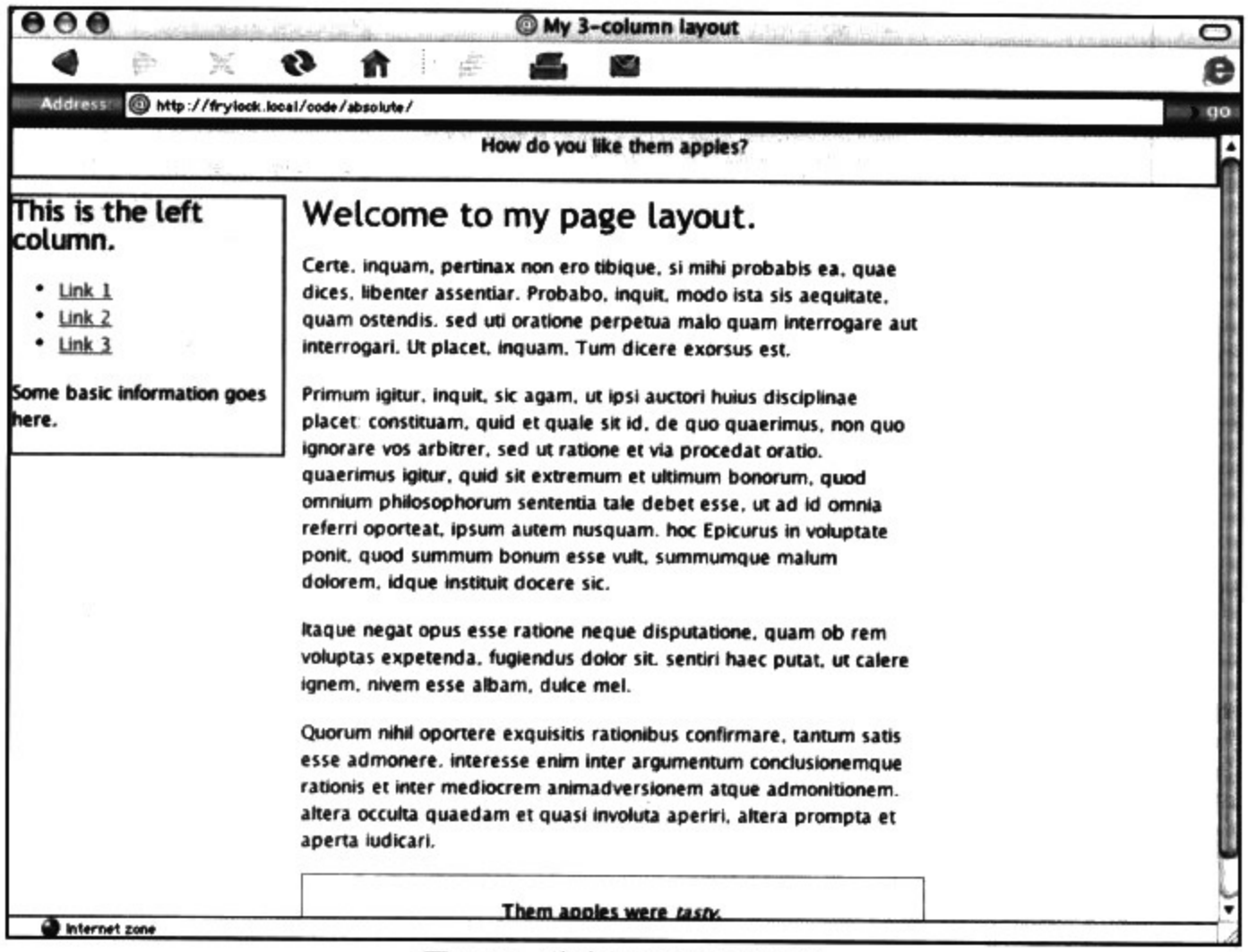


图9-23 将有疑问的地方删除

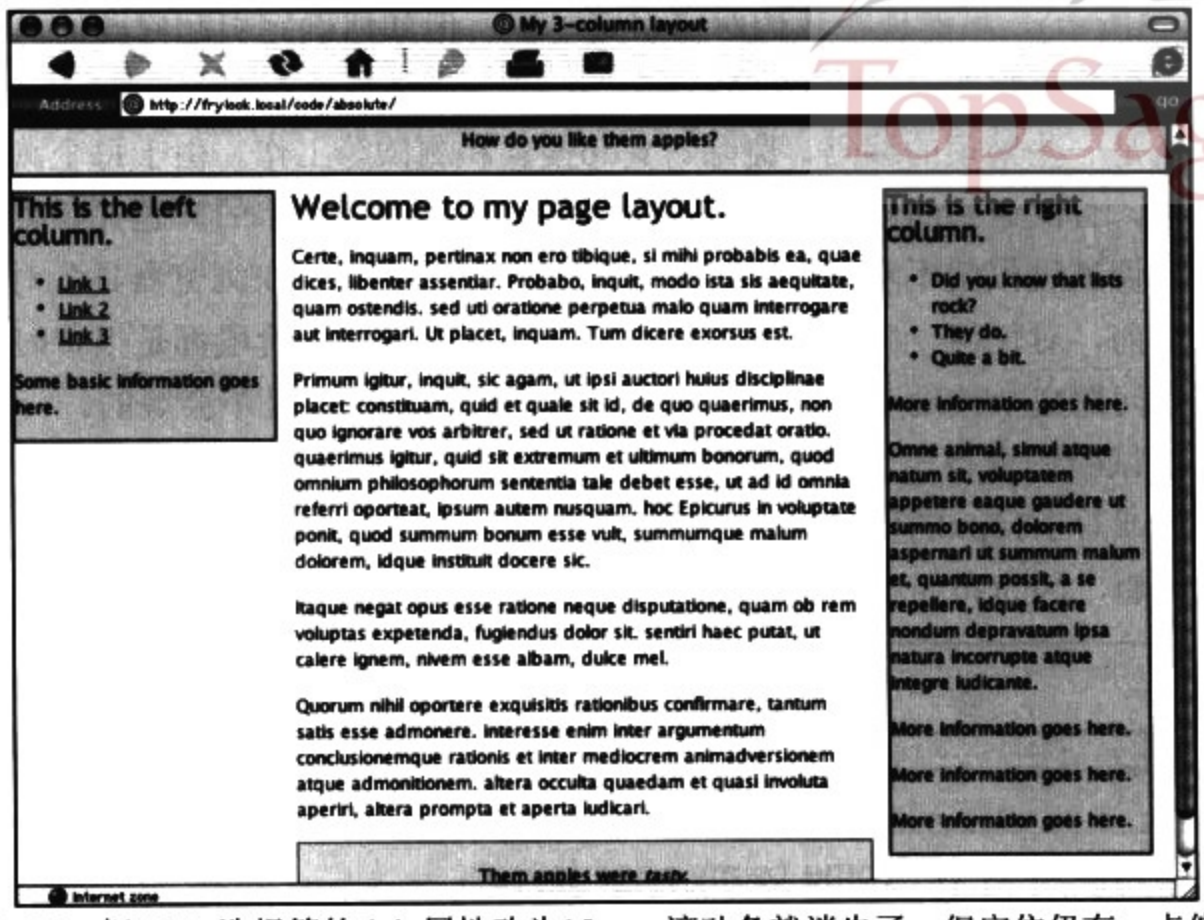


图9-24　把#right选择符的right属性改为15px，滚动条就消失了，但定位仍有一点偏差

因此在删除滚动条后，右边栏不在倚靠窗口的边了。现在应该能修复这个问题了。下面看看还需要进一步做什么：

(1) 即使没有为right div设置外边距，IE 5/Mac似乎强制提供一个“隐藏”的15px外边距。

(2) 因此IE 5/Mac把“margin-right: 0;”看作“margin-right: 15px;”。

由此可以看出，对IE 5/Mac不能是“margin-right: 15px;”，而应翻译为“margin-right: 0;”。#right选择符应该编辑成下列形式：

```
#right {
 position: absolute;
 right: 0;
 top: 0;
 width: 175px;
}
```

现在看看能否应用一些对IE友好的模糊数学：

```
#right {
 position: absolute;
```

```
 margin-right: -15px;
 right: 15px;
 top: 0;
 width: 175px;
}
```

重新加载，看看效果如何(如图9-25所示)。

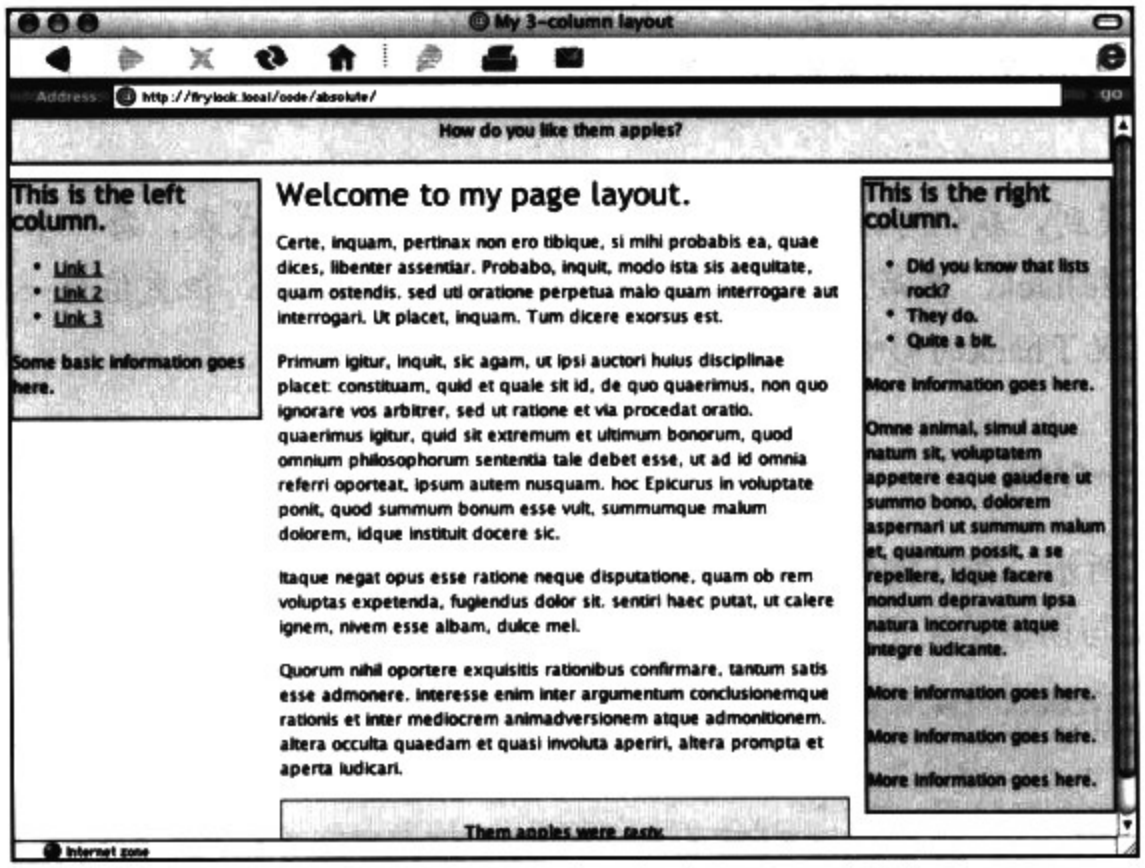


图9-25 应用hack后水平滚动条从IE 5/Mac中消失了

效果非常好。通过迎合IE 5/Mac的呈现怪僻，修复了IE 5/Mac的这个错误。

而且初步测试表明，这个方案并没有对其他表现良好的浏览器带来不好的效果。但为了安全起见，应该把这个hack从实际规则中隔离出来：

```
#right {
 position: absolute;
 top: 0;
 right: 0;
 width: 175px;
}

/*\*//*/
#right {
 margin-right: -15px;
 right: 15px;
```

```
}
/**/
```

第一条规则是最初的#right选择符，这是本章到处都使用的规则。第二条规则包含一两个属性值，用于修复IE 5/Mac的显示问题。这条规则前后围绕的是IE 5/Mac Band Pass Filter(参见第2章)。这些字符序列看起来很怪异，它使第二条规则对除IE 5/Mac外的所有用户代理不可见，保证其他浏览器不受这个hack的影响。

解决了一个错误，下面来看第二个。

说明：

在第2章提到，我们可以创建独立的、与浏览器有关的样式表，每一个包含针对该浏览器特质的一组hack。在第2章介绍了这种方案的好处(和细节)，但更值得一提的是它使核心CSS文件摆脱了hack的影响，因此更易于维护。

9.4.7 IE 5.*x*+/Win

如图9-26所示，在IE的任何Windows版本——IE 5.0、IE 5.5或IE 6.0——中打开测试页，实际结果与期望结果不一致，左边栏应该位于更左一点的位置。

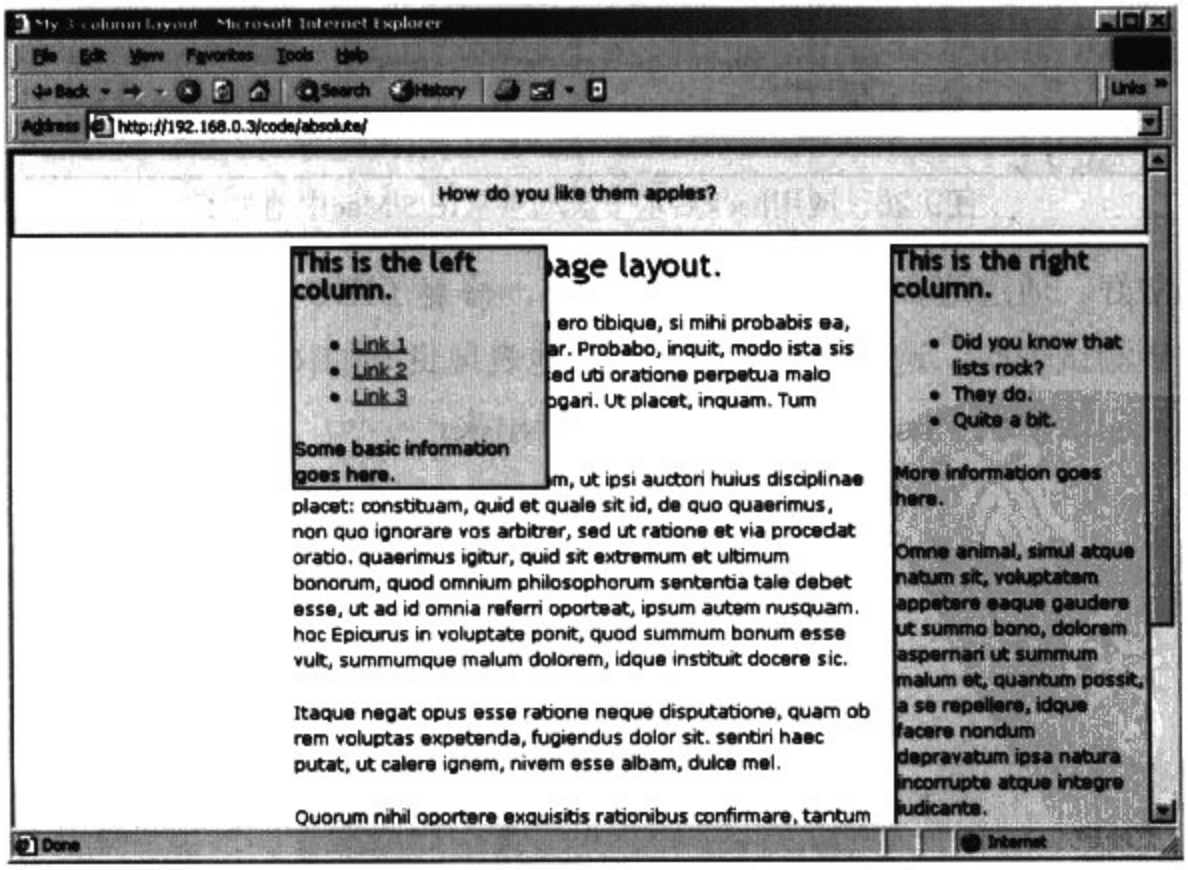


图9-26 现在不是右边而是左边出了问题

刚修复了一个关于#right块的问题，现在又面临一个正相反的问题，或者说是几乎相反。左边栏不是充填在窗口的最左边，而似乎是粘在内容块内。按照解决在IE5/Mac中所遇到问题的方法——删除/修改标记、调整CSS——试着解决这个问题，几个回合下来没有奏效。

值得庆幸的是，在网上的快速搜索得到了一个已知IE/Win错误的信息。在处理一个未声明尺寸的“方块”时(如容器div)，要给出足够空间以容纳其后代元素，按这种方式初次绘制这个“方块”时，IE/Win会出现问题。要解决这个问题，必须告诉IE/Win：“是的，这个“方块”确实有确定的尺寸——如果没什么大问题，希望能正确绘制这个“方块”。”

在最后，一个称为The Holly Hack(http://www.communitymx.com/abstract.cfm?cid=AAA7C45E7CD65D33或 http://positioniseverything.net/articles.html)的、针对IE的解决方案解决了上述问题：

```
/* hide from Mac IE5 \*/
* html #container {
 height: 1%;
}
/* END hide from Mac IE5 */
```

Holly Hack是根据其创建者Holly Bergevin命名的。第一行注释末尾的反斜杠是一个hack，它使IE 5/Mac忽略两个注释行内的所有代码。第二个规则中的选择符以通配选择符(*)开始，后跟html，然后跟一个问题元素的选择符，在这里是容器div。结果Windows和Macintosh操作系统上的IE浏览器都能识别一个被<html>封装的不可见元素。这个hack被称为Star HTML Hack，这种在html选择符前使用通配选择符的用法只对IE浏览器有效。又由于IE 5/Mac不受这个特定布局错误的影响，所以我们在第一行内用一个注释hack使这条规则对这类浏览器是隐藏的。

使用了1%的高度后，现在看看布局是怎么样的(如图9-27所示)。

1%高度规则将对IE错误呈现引擎的启动进行管理。通过给容器块提供一个初始高度(如果很小的话)，IE知道扩展这个块以包含它的所有后代(老实说这一点总应该做到，但不一定非等到由浏览器来做)。

经过这一轮调试后，我们确定三栏布局框架可以很好地用在所测试的所有浏览器。然而，在目前所构造的组件中还遗漏了一个重要的组件。下面我们研究模仿Molly.com所需的最后一个组件，并完成三栏布局的最后设计。

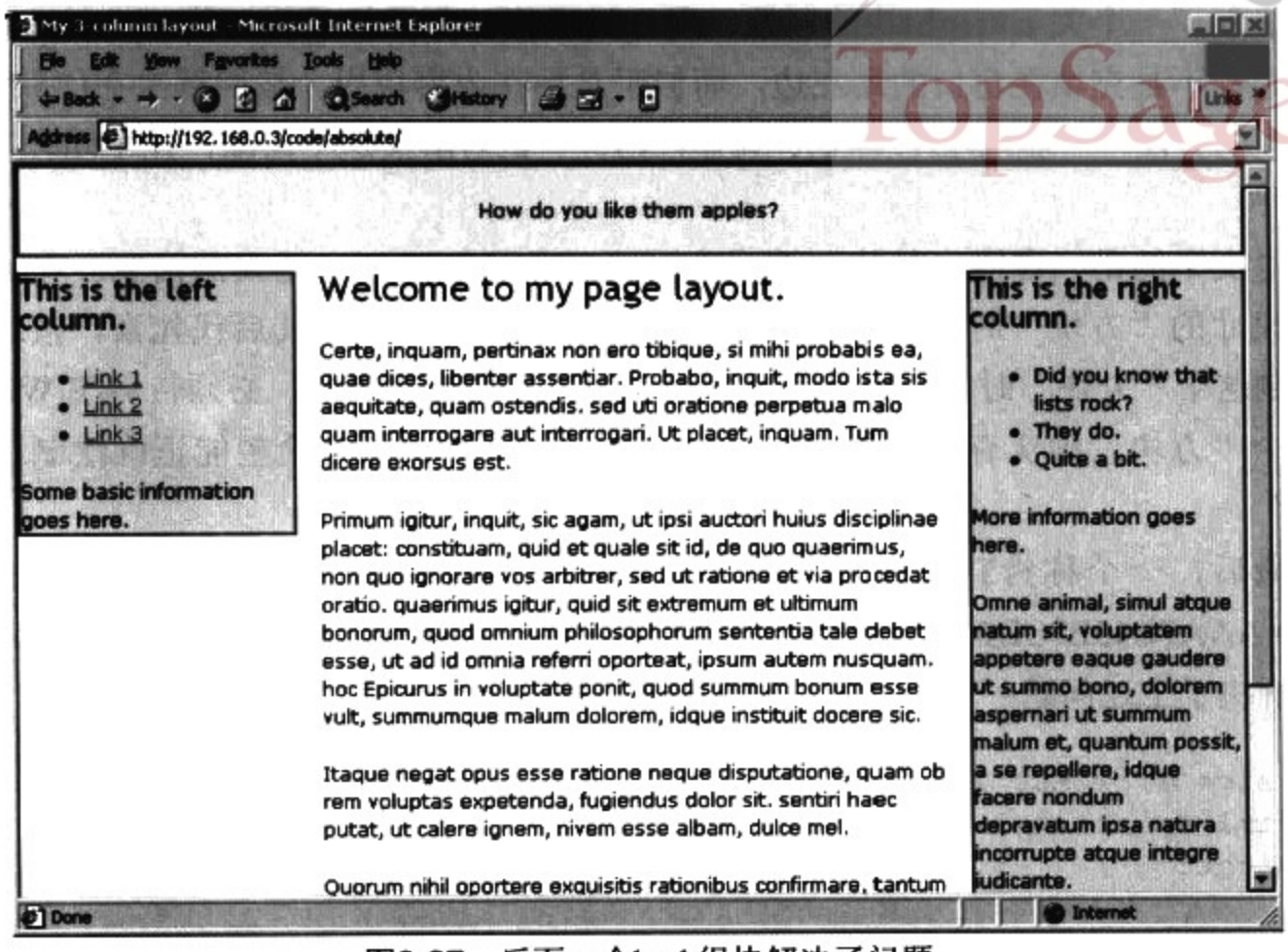


图9-27　后面一个hack很快解决了问题

9.5　设置边界：max-width属性

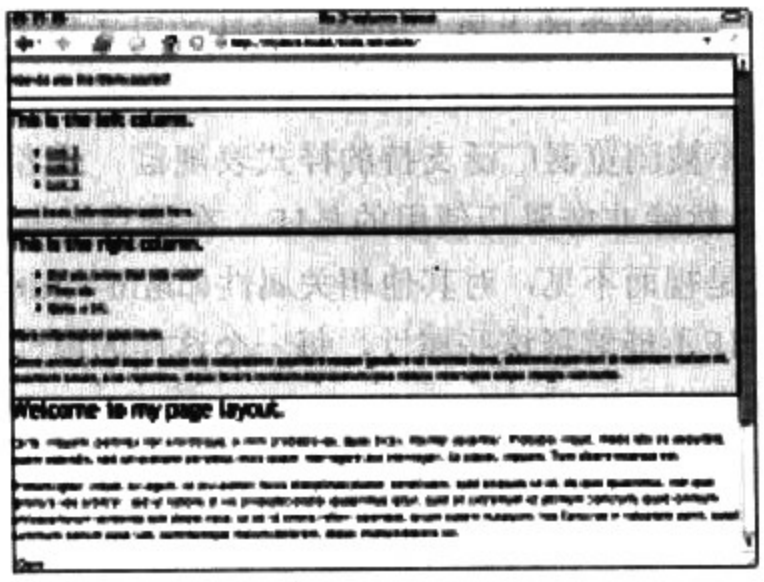
最后一项工作是使设计具有高度灵活的特点。目前的布局扩展到整个浏览器窗口的宽度，而不论它有多窄(或多宽)。在更大一些的窗口里，对于宽度不固定的设计，文本行长度可能变得很长，几乎无法管理，以致很难浏览。

max-width是一个很方便的CSS属性，正如其名称所说的：它为一个给定的元素创建最大宽度。我们在设计中使用这个属性并看看是什么效果。我们把max-width设置为1000px：

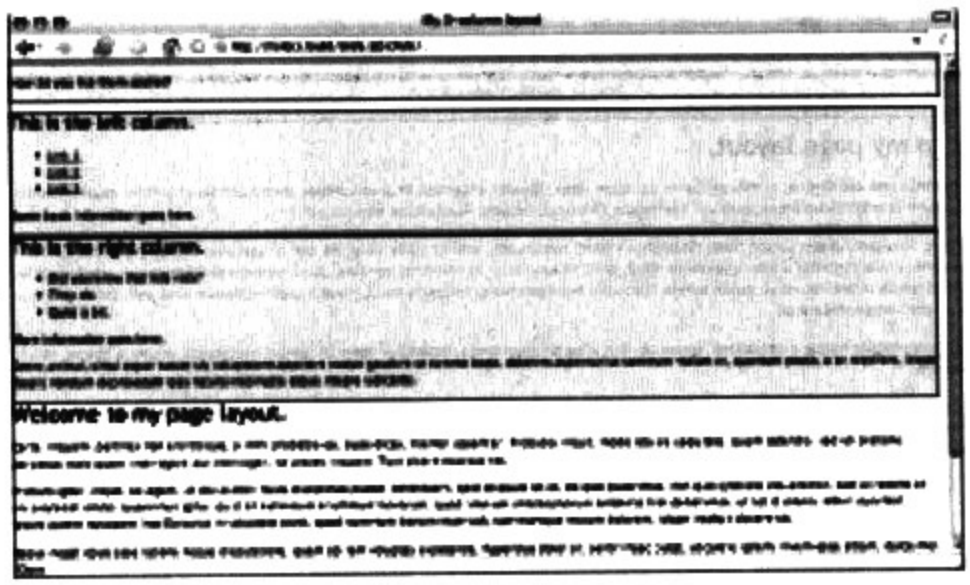
```
#header, #container {
 max-width: 1000px;
}
```

刷新后可能差别并不明显。然而，一旦增加浏览器窗口的宽度，该属性的效果就逐渐变得明显了(如图9-28所示)。

800x600



1024x600



1280x600

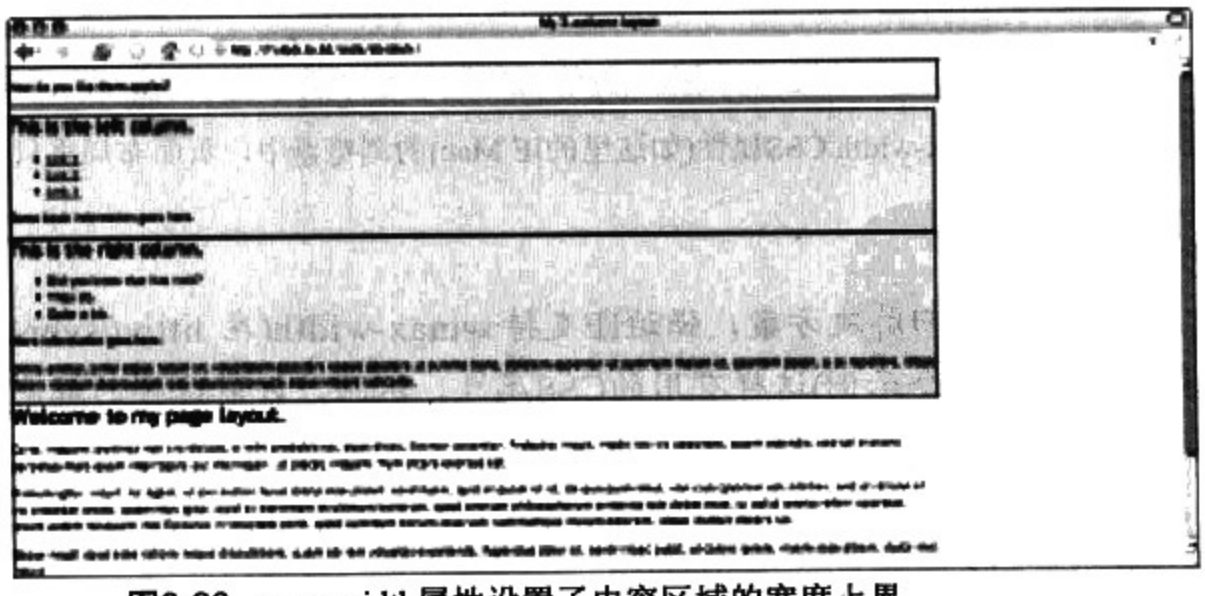


图9-28　max-width属性设置了内容区域的宽度上界

当浏览器窗口放大或缩小时，设计都能进行相应的调整，非常灵活。然而，一旦窗口宽度超过1000px(给max-width属性设置的值)，页面布局就停止缩放了。有了max-width属

性，就给页面的水平宽度设置了一个隐含的上界，以保证设计的灵活程度不妨碍用户快速浏览内容。

不幸的是，max-width也是不被浏览器广泛支持的样式表规范。或者更明确地说，它能得到浏览器的强有力支持，当然除非您碰巧使用的是IE。在撰写本书时，本星球上最流行的浏览器对这个方便属性还是视而不见，对其他相关属性如min-width、min-height和max-height也是视而不见。由于IE不能解释这些属性，每一个这样的属性只能限于理论范围，目前最好的设计并不能依靠它们。

令人欣慰的是，这并不妨碍我们使用这些技术。相反，对于不支持max-width属性的浏览器，我们的设计也能占据整个浏览器的水平空间(如图9-29所示)。

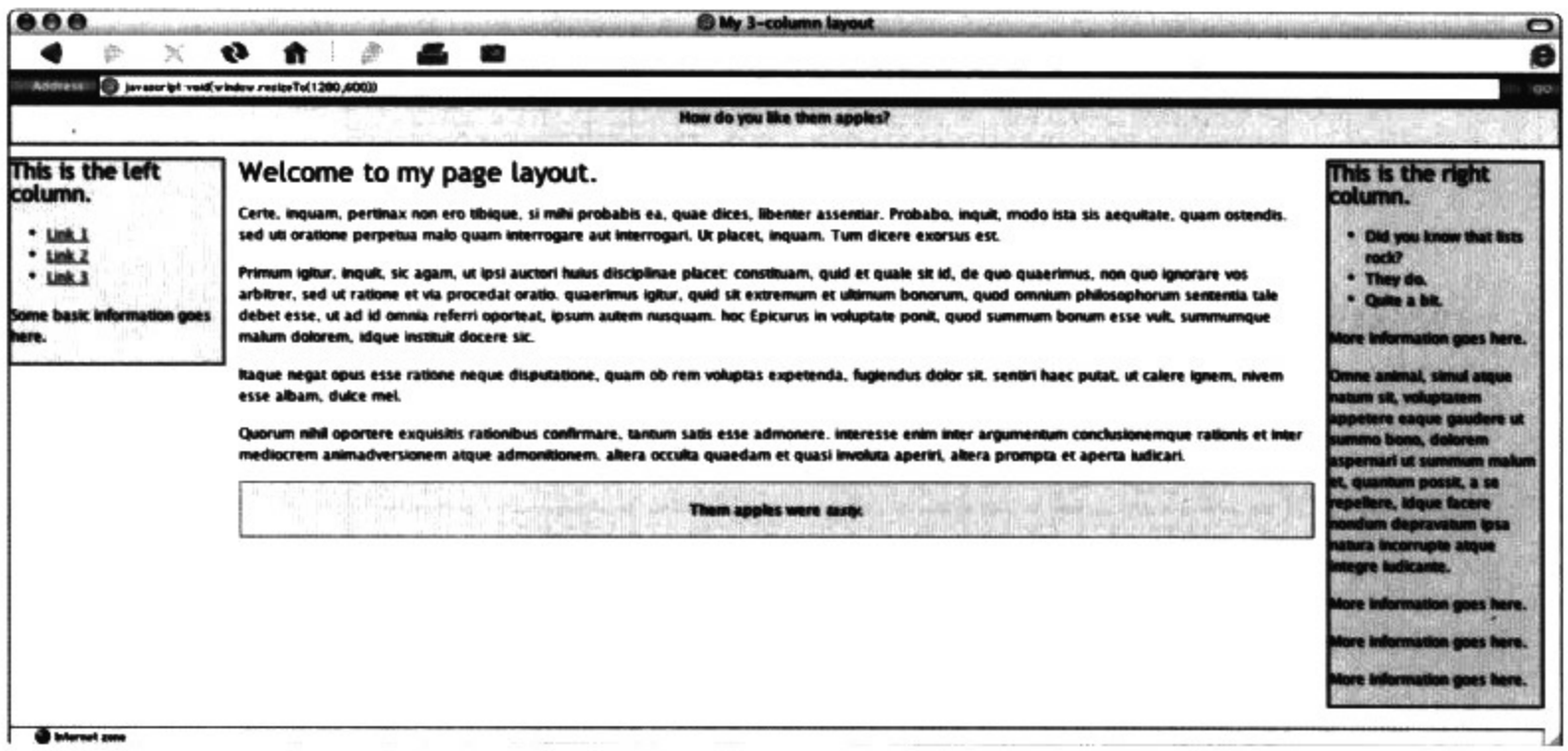


图9-29　在不支持max-width CSS属性(如这里的IE/Mac)的浏览器中，页面布局将只扩展到窗口宽度

说明：

网上有大量的迂回解决方案，强迫IE支持如max-width(在 http://svendtofte.com/code/max_width_in_ie的一个示例)这样有用的CSS属性。然而，许多这样的解决方案把只有IE使用的属性引入到CSS中。要提防的是，这些私有代码可能使代码无效和给其他兼容性更好的浏览器带来付作用。

另外，Dean Edwards编写了“IE7”(http://dean.edwards.name/IE7)，这是一个JavaScript模块库，用于改进IE5以上版本对CSS规范的支持。作为一种只采用JavaScript的方法，Edwards的工作不会使代码无效。不过在用于产品环境时一定要进行充分的测试和评估。

9.6　小结

我们从一个无样式文档的默认正常流开始介绍三栏布局。介绍了如何用CSS定位对默认方案进行重写和如何把元素从文档流中的正常位置删除。

在了解了绝对定位和相对定位后，我们知道了重新构造三栏布局所需的构造块。通过使用包含在相对定位父块内的绝对定位块组合，我们可以创建一个更加灵活的三栏布局。用一个很小的CSS hack创建了一个样式基础。在所有现代浏览器中，这个样式基础看起来都相当智能。用额外的内容和信息可以对它进行充实。

读书笔记

附录A HTML 4.01元素

在用CSS进行设计之前，Web文档的内容必须用HTML元素进行标记。为了有效地使用CSS，必须正确地使用这些HTML元素，要在适当的内容周围使用正确的HTML元素。

表A-1列出了HTML 4.01规范中的所有HTML元素，这是由W3C制定的规范，W3C是制定Web相关标准的管理机构。最左边一列是元素的名称。第二列指明元素是否为起始标签。紧接着的三列对元素进行更详细的描述。列中的O，表示元素的这部分是可选的，F表示禁止，E表示可以为空，D表示建议取消。DTD列提供了元素所属的文档类型定义信息。如果元素只能出现在一类DTD内，则对Loose DTD为L，对Frameset为DTD为F。最后一列是对元素的描述。

表A-1 HTML 4.01中的所有HTML元素

元素名称	起始标签	结束标签	是否可为空	是否建议取消	DTD	描述
A						锚
ABBR						缩写形式(如WWW、HTTP等)
ACRONYM						表示是一个同义词
ADDRESS						作者信息
APPLET				D	L	Java applet
AREA		F	E			客户端图像映射区
B						粗体文本样式
BASE		F	E			文档的基URI
BASEFONT		F	E	D	L	基本的文本字体大小
BDO						I18N BiDi重写
BIG						大型文本样式

(续表)

元素名称	起始标签	结束标签	是否可为空	是否建议取消	DTD	描述
BLOCKQUOTE						大段引用文字
BODY	O	O				文档主体
BR		F	E			强制换行
BUTTON						添加按钮
CAPTION						表格标题
CENTER				D	L	内容居中对齐
CITE						引用
CODE						计算机代码段
COL		F	E			表格列
COLGROUP		O				表格列组
DD		O				定义描述
DEL						被删除的文本
DFN						实例定义
DIR				D	L	目录列表
DIV						一个区域块
DL						定义列表
DT		O				定义性术语
EM						强调
FIELDSET						表单控件组
FONT				D	L	字体的本地修改
FORM						交互式表单
FRAME		F	E		F	子窗口
FRAMESET					F	框架容器；框架体的替代元素
H1						第一级标题
H2						第二级标题
H3						第三级标题
H4						第四级标题
H5						第五级标题
H6						第六级标题

(续表)

元素名称	起始标签	结束标签	是否可为空	是否建议取消	DTD	描述
HEAD	O	O				文档头
HR		F	E			水平标尺
HTML	O	O				文档根元素
I						斜体文本样式
IFRAME					L	内联子窗口
IMG		F	E			内嵌图像
INPUT		F	E			表单控件
INS						插入到文档中的文本
ISINDEX		F	E	D	L	单行提示
KBD						用户输入的文本
LABEL						表单字段标签文本
LEGEND						为Fieldset元素指定标题
LI		O				列表项
LINK		F	E			一个独立于媒介的链接
MAP						客户端图像映射
MENU				D	L	菜单列表
META		F	E			一般元信息
NOFRAMES					F	当框架结构不被支持时，显示替换内容的容器
NOSCRIPT						当一种语言不被支持时，显示替换内容的容器
OBJECT						通用嵌入对象
OL						有序列表
OPTGROUP						选项组
OPTION		O				可选项
P		O				段落
PARAM		F	E			有名属性值
PRE						预先被格式化的文本

(续表)

元素名称	起始标签	结束标签	是否可为空	是否建议取消	DTD	描述
Q						简短内联引用
S				D	L	给文本加上删除线样式
SAMP						提取程序输出结果、脚本等
SCRIPT						脚本声明
SELECT						选项选择符
SMALL						小文本样式
SPAN						通用语言/一个内联样式容器
STRIKE				D	L	带删除线的文本
STRONG						着重强调
STYLE						样式信息
SUB						下标
SUP						上标
TABLE						表格容器
TBODY	O	O				表格实体
TD		O				表格数据单元格
TEXTAREA						多行文本字段
TFOOT		O				表格页脚
TH		O				表格页眉单元格
THEAD		O				表格页眉
TITLE						文档标题
TR		O				表格行
TT						等宽字型文本样式
U				D	L	下划线文本样式
UL						无序列表
VAR						变量或程序参数实例

HTML4.01元素列表(http://www.w3.org/TR/html4/index/elements.html)是W3C(马萨诸塞技术研究所、欧洲信息与数学研究会、Keio大学) 于1999年12月24日推出的。版权所有。http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231。

附 B 录

从HTML到XHTML的转换规则

超文本标记语言(HTML)是一种简单的语言，它使Web在20世纪90年代繁荣起来。然而其简单性也是进步的障碍。HTML的早期成功吸引了更多的Web开发人群并推动媒体的发展。HTML为简单付出了代价。

例如，虽然把一个图像放到Web页面用HTML实现起来非常容易，但把图像放到Web页面指定位置不可能不使用table标签。另一个例子是在Web页面中放一些多媒体内容，这常常导致要使用一些无效的、专有的元素和属性。

另外，HTML包含一个有限的元素和属性集。如工程或化工等其他行业不能标记它们的公式。W3C不是采取制定一个能包容一切的HTML版本，而是致力于扩展标记语言(XML)，这是一种灵活的元语言。

XML提供了一个创建其他标记语言的框架。其他行业可以创建自己的标记语言而不是受限于如HTML这样的环境。

然而，对于大多数起初就熟悉HTML的Web开发人员来说，XML的主要好处(创建一个新元素并指定处理方法)并不重要。相反，HTML中的元素将非常有用。

W3C从XML标准重新对HTML进行了修订以实现向后兼容，使这种语言体现了XML中的结构。XHTML的本质就是按XML语法定义的HTML。换句话说，XHTML是一个严格的指导方针集合，使熟悉HTML的Web开发人员完全不费劲地就能编写出有效的XML文档。

然而，当开发人员来到一个更严格的编程环境时，从HTML到XHTML的内容重构是一个非常头疼的工作。和老式的HTML和浏览器相比，XHTML语法(或编码规则)更不能容忍编码错误。

为了帮助大家对如何正确编写XHTML有更深入的了解，本附录起到一个把Web开发

人员从老式的HTML开发人员转变为彻底的XHTML用户的指南作用。

B.1 XML声明

无毫疑问，Web开发人员肯定知道位于Web文档顶端的html元素的重要性。对于XHTML，可在html元素之上添加如下所示的行：

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

这一行仅仅声明我们使用的是XHTML version1.0，字符集为iso-8859-1。

XML声明是推荐使用的，但不是必需的。由于它位于Web文档顶端且只有一行，为什么不包含它呢？下面是使用XML声明时可能出现的问题。

- 当浏览一个Web页面的源码而不是呈现文档时，一些浏览器可能呈现的只是标记。
- 其他一些浏览器可能把这个Web文档解析为一个XML树而不是呈现出文档。
- 在IE 6.0/Win中，浏览器将以非常怪异的模式显示Web文档，即使Web文档是有效的。
- 如果用PHP创建动态页面，您可能注意到，这一行开始的左尖括号和问号与开始编写PHP代码时一样。如果把这行代码留在PHP文档中，就会使服务器引起混乱，它不会成功地解析您的页面。这种情形的迂回解决方案是在文档的开始处用PHP的echo函数写出第一行：

```
<?php echo "<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>\n"; ?>
```

B.2 选择合适的级别

XHTML有三种级别：严格级、过渡级和框架级。这是根据三种不同DTD而分类的。DTD定义XHTML和确定哪些元素和属性可用，以及如何使用。可以把一个DTD看成是某个文档允许使用的术语字典。

要创建一个有效的XHTML文档，必须包含一个DOCTYPE声明，这个声明组成了位于文档顶部且在XML声明之下的一行或两行。这行代码指明所用的DTD类型，并指示浏览器和有效性验证软件如何处理该文档的内容。

如果把Web文档定义为严格级，意味着不仅要遵守规则条文而且要领会规则的实质。您将成为一个真正的XHTML信奉者，以后不再把任何HTML元素用于页面的表现。对于严格的DTD，只能用HTML元素标记内容，不能对表现形式进行控制。把下列代码放在

XML声明之后、HTML声明之前：

```
<!DOCTYPE html
 PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

说明：

可能小写字母比较难读，用大写是XHTML1而不是XHTM11。

如果要研究XHTML但又希望能使用一些建议取消的元素和属性，或希望能使用一些经典的HTML标签，则过渡DTD是最好的选择。

```
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

框架级DTD是针对在Web页面中要求使用框架的Web文档：

```
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

框架级DTD只用在包含框架的文档中。不必对每个Web文档使用框架级DTD，Web文档组成了框架集的一个框架。对于这些文档，我们可以使用严格的DTD或过渡的DTD。

B.3　XHTML规则

既然我们已经设置了XML声明和DTD，下一步就是如何正确格式化Web文档。下面几部分涉及如何对内容进行正确标记、如何正确使用XHTML。

B.3.1　不要忘记命名空间属性

在文档顶部声明所用的文档类型，这个声明指明在文档中可以使用的元素和属性。与DOCTYPE声明一起，命名空间是确定文档标记的另外一种手段，这里是XHTML。

要确定命名空间，在html元素的起始html标签中添加一个命名空间属性xmlns：

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
```

B.3.2　引用属性值

一个元素的所有属性值都要求用引号括起来。因此不能使用下面的代码：

```
<img src=file.gif width=133 height=133 />
```

正确的代码是：

```
<img src="file.gif" width="133" height="133" />
```

B.3.3 属性不能最小化

HTML中的一些元素(如水平尺标签hr)，其属性可以最小化，只列出属性名是有效的：

```
<hr noshade />
```

然而在XHTML中就不能对属性最小化。当遇到一个从不要求值的属性时，要把值设置为属性名。我们以hr元素为例，noshade属性的值被设置为noshade：

```
<hr noshade="noshade" />
```

B.3.4 终止空元素

空元素是那些不成对出现的元素(如img、br或hr)。

在HTML中非空元素非常普遍(如p或h2)。它们用于标记Web页面中内容的开始和结束。p标签表示的是一个段落，如下例所示：

```
<p>That's when I thought I should decline a second helping of her
infamous
spaghetti and meatball salad.</p>
```

对于XHTML，所有元素(包括空元素)必须被终止。

要在XHTML继续使用空元素，必须对空元素做小小的修改。在元素的末尾添加一个空格和一个斜杠：

```
<img src="file.gif" alt="Company logo" width="125" height="36" />
```

注意，在斜杠前添加一个空格对代码有效性来讲并不是必需的，但这可以防止老式浏览器(如Netscape Navigator 4)呈现这个元素时出现错误。

B.3.5 清晰嵌套

正确嵌套元素非常简单，并已成为任意一个Web开发人员实践的一部分。在下列行中，strong元素的结束标签位于p元素结束标签之外。

```
    <p>That's when I thought I should <strong>decline a second helping of
her infamous
    spaghetti and meatball salad.</p></strong>
```

然而标记内容的正确方法是：

```
    <p>That's when I thought I should <strong>decline</strong> a second
helping of her infamous spaghetti and meatball salad.</p>
```

B.3.6　包含CSS和JavaScript文件的XHTML

结合CSS和JavaScript文件是体现Web页面表现和行为的首选方法：

```
    <script src="/js/validator.js" type="text/javascript"></script>
    <link rel="stylesheet" href="/css/layout.css" type="text/css" />
```

如果要使用内部JavaScript，可用起始标记<![CDATA[和结束标记]]>把代码封装起来。

```
    <script type="text/javascript">
    /* <![ CDATA[ */
    // Javascript goes here
    /* ]]> */
    </script>
```

B.3.7　小写元素和属性名

XHTML中的所有元素和属性必须设置为小写，不能设置为大写或大小写混合。下面示例是不正确的用法：

```
<HTML> </HTML>
<Strong></Strong>
```

下列示例是正确的用法：

```
<body> </body>
```

当然大小写混合的属性值仍然是有效的：

```
<a href="IWantToBelieve.html">Photos of Aliens</a>
```

B.3.8 使用name时引入ID

在XHTML中，name是建议取消的属性，将来会从规范中删除。取代它的是id属性。在name属性不再是有效属性之前，在name属性后附加id属性：

```
<a name="admin" id="admin">Administration at CLC</a>
```

B.3.9 对&编码

当一个属性值中包含&符号时，要确保使用的是该字符实体，即&amp。

在对&符号编码时和与动态页面打交道时，通过URL串传给浏览器的参数如下所示：

```
<a href="add-cart.html?isbn=9780470177082&amp;id=023">Add this item to your cart</a>
```

B.3.10 有疑问时进行有效性验证

我们都是人，编码时都会出现错误。为了帮助指出XHTML的问题，或仅是为了保证所编写代码是正确的，可以把页面交给如http://validator.w3.org这样的验证软件进行测试。

在验证软件中还内置了大多数WYSIWYG 和一些非WYSIWYG的Web编著工具。可阅读软件附带的文档对验证软件进行更多了解。

附录C CSS 2.1属性

在用HTML标记元素时，必须对正在处理的元素有所了解。用CSS进行设计时同样如此，只有充分了解属性和它们的值才能进行高效的Web设计。

在本附录的表C-1中列出了所遇到的所有CSS 2.1属性。最左边一列是CSS属性名。接下来两列是属性的所有可能值和初始值。第四列说明CSS属性所适用的HTML元素。第五列说明这个属性能否被其他元素继承。最右边一列说明适用的媒介组。

说明：

CSS 2.1属性列表(www.w3.org/TR/CSS21/propidx.html) 是W3C (马萨诸塞技术研究所、欧洲信息与数学研究会、 Keio大学) 于2004年2月25日推出的，如表C-1所示。版权所有。http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231。

表C-1 CSS 2.1属性列表

属性名	值	初始值	应用范围 (默认：所有元素)	能否被继承	媒介组
'azimuth'	<angle> | [[left-side | far-left | left | center-left | center | center-right | right | far-right | right-side] || behind] | leftwards | rightwards | inherit	center	所有元素	能被继承	音频
'background-attachment'	scroll | fixed | inherit	scroll	所有元素	不能被继承	视频
'background-color'	<color> | transparent | inherit	transparent	所有元素	不能被继承	视频
'background-image'	<uri> | none | inherit	none	所有元素	不能被继承	视频

(续表)

属性名	值	初始值	应用范围 (默认：所有元素)	能否 被继承	媒介组
'background-position'	[[<percentage> | <length> | left| center | right] [<percentage> |<length> | top | center | bottom]?] | [[left | center | right] ||[top | center | bottom]] |inherit	0% 0%	所有元素	不能被继承	视频
'background-repeat'	repeat | repeat-x | repeat-y |no-repeat | inherit	repeat	所有元素	不能被继承	视频
'background'	[' background-color' || ' backgroundimage' || ' background-repeat' ||' background-attachment' ||' background-position'] | inherit	速记属性见单独的属性	所有元素	不能被继承	视频
'border-collapse'	collapse | separate | inherit	separate	'table'和'inline-table'元素	能被继承	视频
'border-color'	[<color> | transparent]{1,4} |inherit	速记属性，见单独的属性	所有元素	不能被继承	视频
'border-spacing'	<length> <length>? | inherit	0	'table'和'inline-table'元素	能被继承	视频
'border-style'	<border-style>{1,4} | inherit	速记属性，见单独的属性	所有元素	不能被继承	视频
'border-top' 'border-right" border bottom' 'border-left'	[<border-width> || <border-style>|| 'border-top-color '] | inherit	速记属性，见单独的属性	所有元素	不能被继承	视频
'border-top-color' 'border-right-color' 'border-bottom-color' 'border-left-color'	<color> | transparent | inherit	color属性的值	所有元素	不能被继承	视频
'border-top-style' 'border-right-style' 'border-bottom-style' 'border-left-style'	<border-style> | inherit	none	所有元素	不能被继承	视频
'border-top-width' 'border-right-width' 'border-bottom-width' 'border-left-width'	<border-width> | inherit	medium	所有元素	不能被继承	视频

(续表)

属性名	值	初始值	应用范围 (默认：所有元素)	能否被继承	媒介组
'border-width'	<border-width>{1,4} | inherit	速记属性，见单独的属性	所有元素	不能被继承	视频
'border'	[<border-width> || <border-style> || 'border-top-color'] | inherit	速记属性，见单独的属性	所有元素	不能被继承	视频
'bottom'	<length> | <percentage> | auto | inherit	auto	已定位的元素	不能被继承	视频
'caption-side'	top | bottom | inherit	top	'table-caption'元素	能被继承	视频
'clear'	none | left | right | both | inherit	none	块级元素	不能被继承	视频
'clip'	<shape> | auto | inherit	auto	绝对定位的元素	不能被继承	视频
'color'	<color> | inherit	取决于用户代理	所有元素	能被继承	视频
'content'	normal | [<string> | <uri> | <counter> | attr(<identifier>) | open-quote | close-quote | no-open-quote | no-close-quote]+ | inherit	normal	:before和:after伪元素	不能被继承	所有
'counter-increment'	[<identifier> <integer>?]+ | none | inherit	none	所有元素	不能被继承	所有
'counter-reset'	[<identifier> <integer>?]+ | none | inherit	none	所有元素	不能被继承	所有
'cue-after'	<uri> | none | inherit	none	所有元素	不能被继承	音频
'cue-before'	<uri> | none | inherit	none	所有元素	不能被继承	音频
'cue'	['cue-before' || 'cue-after'] | inherit	速记属性，见单独的属性	所有元素	不能被继承	音频
'cursor'	[[<uri> ,]* [auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize | text | wait | help | progress]] | inherit	auto	所有元素	能被继承	视频，交互式媒介
'direction'	ltr | rtl | inherit	ltr	所有元素	能被继承	视频

(续表)

属性名	值	初始值	应用范围 (默认：所有元素)	能否被继承	媒介组
'display'	inline | block | list-item | run-in | inline-block | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit	inline	所有元素	不能被继承	所有
'elevation'	<angle> | below | level | above | higher | lower | inherit	level	toble	能被继承	音频
'empty-cells'	show | hide | inherit	show	'toble-cell '元素	能被继承	视频
'float'	left | right | none | inherit	none	除已定位元素和生成的内容之外的所有元素	不能被继承	视频
'font-family'	[[<family-name>| <generic-family>] [, <family-name> |<generic-family>]*] | inherit	取决于用户代理	所有元素	能被继承	视频
'font-size'	<absolute-size> | <relative-size> | <length> | <percentage> | inherit	medium	所有元素	能被继承	视频
'font-style'	normal | italic | oblique | inherit	normal	所有元素	能被继承	视频
'font-variant'	normal | small-caps | inherit	normal	所有元素	能被继承	视频
'font-weight'	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	normal	所有元素	能被继承	视频
'font'	[['font-style' || 'font-variant'|| 'font-weight']? 'font-size' [/'line-height']? 'font-family']| caption | icon | menu | message-box | small-caption | status-bar |inherit	速记属性，见单独的属性	所有元素	能被继承	视频
'height'	<length> | <percentage> | auto | inherit	auto	除不可替换的内联元素、表格列和列组之外的所有元素	不能被继承	视频

(续表)

属性名	值	初始值	应用范围 (默认：所有元素)	能否被继承	媒介组
'left'	<length> | <percentage> | auto | inherit	auto	已定位的元素	不能被继承	视频
'letter-spacing'	normal | <length> | inherit	normal	所有元素	能被继承	视频
'line-height'	normal | <number> | <length> | <percentage> | inherit	normal	所有元素	能被继承	视频
'list-style-image'	<uri> | none | inherit	none	带′ display: list-item′ 的元素	能被继承	视频
'list-style-position'	inside | outside | inherit	outside	带′ display: list-item′ 的元素	能被继承	视频
'list-style-type'	disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin| upper-latin | armenian |georgian | none | inherit	disc	带′ display: list-item′ 的元素	能被继承	视频
'list-style'	['list-style-type' || 'list-style-position'|| 'list-style-image'] |inherit	速记属性，见单独的属性	带′ display: list-item′ 的元素	能被继承	视频
'margin-right' 'margin-left'	<margin-width> | inherit	0	除带表格显示类型(而不是table和inline-table)的元素之外的所有元素	不能被继承	视频
'margin-top' 'margin-bottom'	<margin-width> | inherit	0	除带表格显示类型(而不是table和inline-table)的元素之外的所有元素	不能被继承	视频
'margin'	<margin-width>{1,4} | inherit	速记属性，见单独的属性	除带表格显示类型(而不是table和inline-table)的元素之外的所有元素	不能被继承	视频
'max-height'	<length> | <percentage> | none | inherit	none	除不可替换的内联元素、表格元素之外的所有元素	不能被继承	视频
'max-width'	<length> | <percentage> | none | inherit	none	除不可替换的内联元素、表格元素之外的所有元素	不能被继承	视频

(续表)

属性名	值	初始值	应用范围 (默认：所有元素)	能否 被继承	媒介组
'min-height'	<length> | <percentage> | inherit	0	除不可替换的内联元素、表格元素之外的所有元素	不能被继承	视频
'min-width'	<length> | <percentage> | inherit	0	除不可替换的内联元素、表格元素之外的所有元素	不能被继承	视频
'orphans'	<integer> | inherit	2	块级元素	能被继承	视频，分页媒介
'outline-color'	<color> | invert | inherit	invert	所有元素	不能被继承	视频，交互式媒介
'outline-style'	<border-style> | inherit	none	所有元素	不能被继承	视频，交互式媒介
'outline-width'	<border-width> | inherit	medium	所有元素	不能被继承	视频，交互式媒介
'outline'	['outline-color' || 'outline-style' || 'outline-width'] | inherit	速记属性，见单独的属性	所有元素	不能被继承	视频，交互式媒介
'overflow'	visible | hidden | scroll | auto | inherit	visible	块级替换元素、表格单元、内联块	不能被继承	视频
'padding-top' 'padding-right' 'padding-bottom' 'padding-left'	<padding-width> | inherit	0	除带表格显示类型(而不是table、inline-table和table-cell)的元素之外的所有元素	不能被继承	视频
'padding'	<padding-width>{1,4} | inherit	速记属性，见单独的属性	除带表格显示类型(而不是table、inline-table和table-cell)的元素之外的所有元素	不能被继承	视频
'page-break-after'	auto | always | avoid | left | right | inherit	auto	块级元素	不能被继承	视频，分页媒介
'page-break-before'	auto | always | avoid | left | right | inherit	auto	块级元素	不能被继承	视频，分页媒介
'page-break-inside'	avoid | auto | inherit	auto	块级元素	能被继承	视频，分页媒介
'pause-after'	<time> | <percentage> | inherit	0	所有元素	不能被继承	音频
'pause-before'	<time> | <percentage> | inherit	0	所有元素	不能被继承	音频

(续表)

属性名	值	初始值	应用范围 (默认：所有元素)	能否被继承	媒介组
'pause'	[[<time> | <percentage>] {1,2}] | inherit	速记属性，见单独的属性	所有元素	不能被继承	音频
'pitch-range'	<number> | inherit	50	所有元素	能被继承	音频
'pitch'	<frequency> | x-low | low | medium | high | x-high | inherit	medium	所有元素	能被继承	音频
'play-during'	<uri> [mix || repeat]? | auto | none | inherit	auto	所有元素	不能被继承	音频
'position'	static | relative | absolute | fixed | inherit	static	所有元素	不能被继承	视频
'quotes'	[<string> <string>]+ | none | inherit	取决于用户代理	所有元素	能被继承	视频
'richness'	<number> | inherit	50	所有元素	能被继承	音频
'right'	<length> | <percentage> | auto | inherit	auto	已定位的元素	不能被继承	视频
'speak-header'	once | always | inherit	once	有表头信息的元素	能被继承	音频
'speak-numeral'	digits | continuous | inherit	continuous	All	能被继承	音频
'speak-punctuation'	code | none | inherit	none	All	能被继承	音频
'speak'	normal | none | spell-out | inherit	normal	All	能被继承	音频
'speech-rate'	<number> | x-slow | slow | medium | fast | x-fast | faster | slower | inherit	medium	All	能被继承	音频
'stress'	<number> | inherit	50	All	能被继承	音频
'table-layout'	auto | fixed | inherit	auto	'table'和'inline-table'元素	不能被继承	视频
‘text-align’	left | right | center | justify | inherit	'left' if 'direction' is 'ltr'; 'right' if'direction'is 'rtl'	块级元素、表格单元和内联块	能被继承	视频
'text-decoration'	none | [underline || overline ||line-through || blink] | inherit	none	所有元素	不能被继承	视频

(续表)

属性名	值	初始值	应用范围 (默认：所有元素)	能否被继承	媒介组
'text-indent'	<length> | <percentage> | inherit	0	块级元素、表格单元和内联块	能被继承	视频
'text-transform'	capitalize | uppercase | lowercase |none | inherit	none	所有元素	能被继承	视频
'top'	<length> | <percentage> | auto | inherit	auto	已定位的元素	不能被继承	视频
'unicode-bidi'	normal | embed | bidi-override |inherit	normal	所有元素	不能被继承	视频
'vertical-align'	baseline | sub | super | top |text-top | middle | bottom | text-bottom| <percentage> | <length> |inherit	baseline	内联级和table-cell元素	不能被继承	视频
'visibility'	visible | hidden | collapse | inherit	visible	所有元素	能被继承	视频
'voice-family'	[[<specific-voice> | <generic-voice>],]* [<specific-voice> |<generic-voice>] | inherit	取决于用户代理	所有元素	能被继承	音频
'volume'	<number> | <percentage> | silent |x-soft | soft | medium | loud |x-loud | inherit	medium	所有元素	能被继承	音频
'white-space'	normal | pre | nowrap | pre-wrap |pre-line | inherit	normal	所有元素	能被继承	视频
'widows'	<integer> | inherit	2	块级元素	能被继承	视频,分页媒介
'width'	<length> | <percentage> | auto | inherit	auto	除不可替换的内联元素、表格列和列组之外的所有元素	不能被继承	视频
'word-spacing'	normal | <length> | inherit	normal	所有元素	能被继承	视频
'z-index'	auto | <integer> | inherit	auto	已定位的元素	不能被继承	视频

附录D

CSS故障排除指南

即使盯着显视器看了几小时，代码看上去相当出色，但单击刷新按钮是不是就出问题了？

CSS初学者和大师可能同样经历过类似的情况。本故障排除指南能解决一些问题。

D.1 有效性

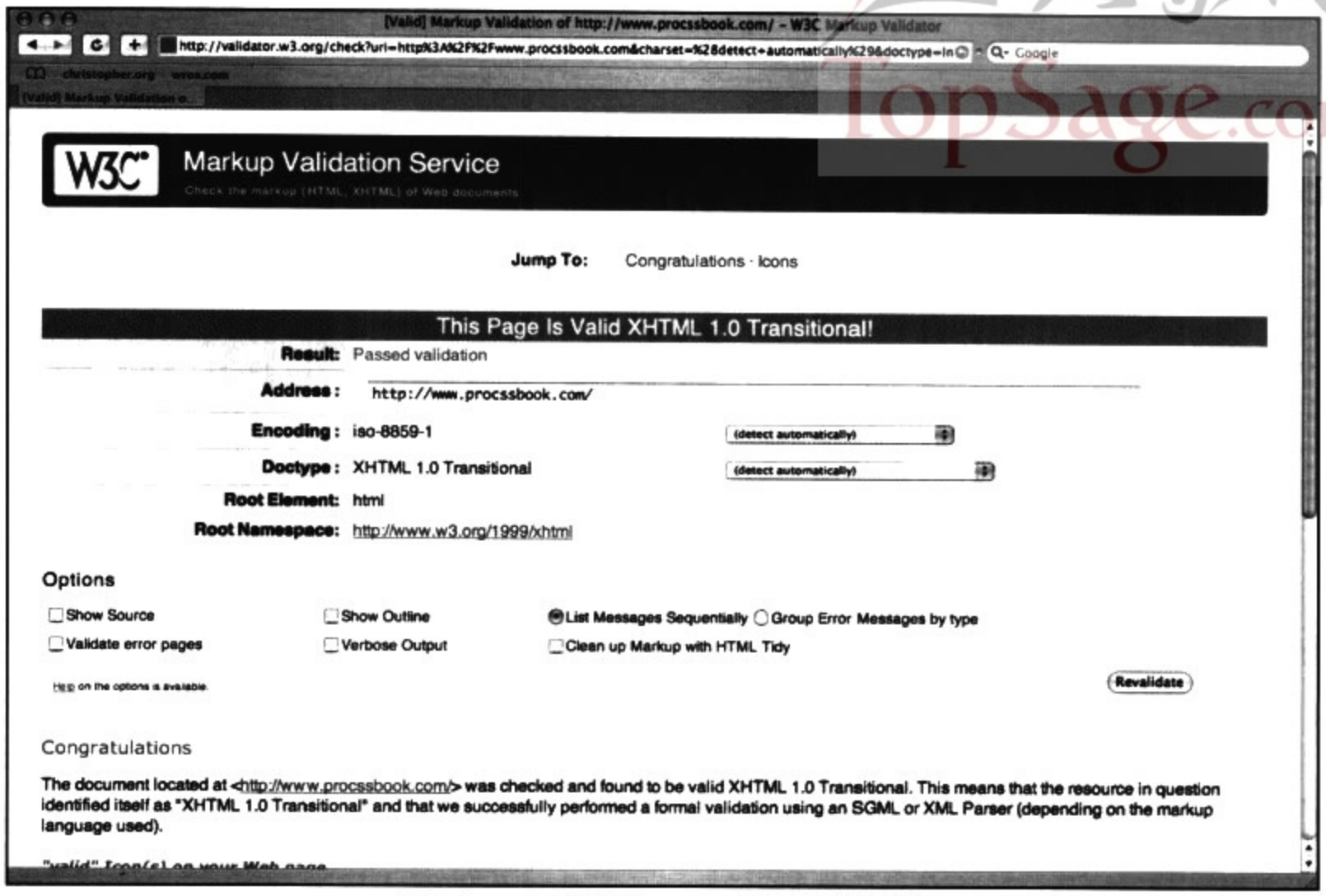
遇到问题时首先要考虑的是HTML和CSS语法是否正确。Adobe Dreamweaver或Microsoft Expression生成的语法可能会隐藏一些代码，因此对设计仍然需要检查。

如果Web开发软件本身并没有带自己的有效性验证器(仔细检查一下软件文档)，要确保设置了优先选项使Web开发软件能排除专有元素(如center)，这样有效验证软件将对标准DTD进行检查。

下面将介绍对HTML和CSS进行有效性验证的Web站点。

D.1.1 HTML

HTML有效性验证服务如图D-1所示，网址为http://validator.w3.org。



图D-1　W3C提供的HTML有效性验证服务

一旦来到该站点，在表单中输入出现问题的页面URL。如果使用URL，要确保Web地址在Web上是可访问的，即这个文件不能位于防火墙之后，或位于一个使用密码的保护区(如内部网)。如果HTML位于上述某个区，就使用有效性验证服务提供的上传功能。

说明：

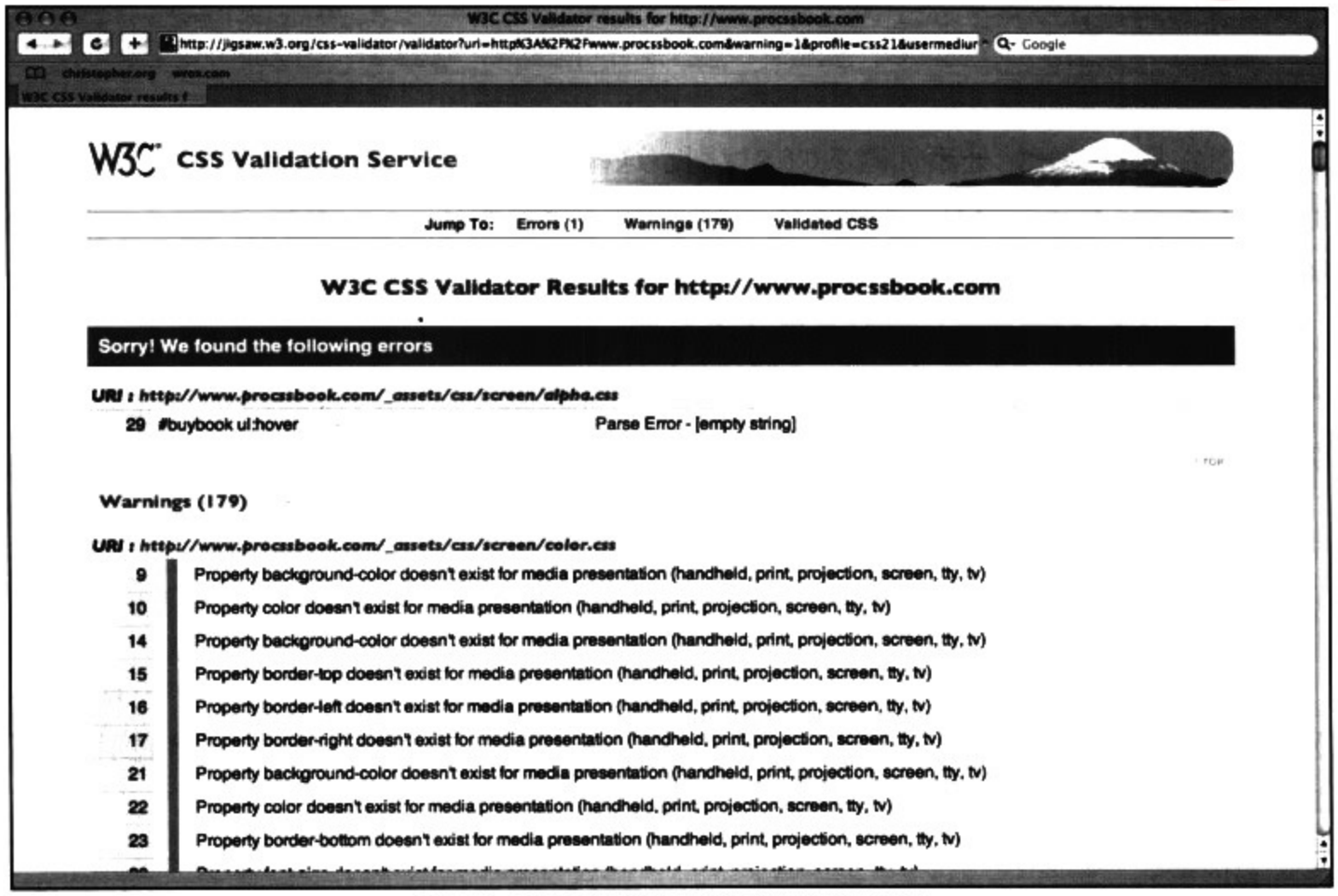
对于HTML元素请参见附录A。如果需要了解从HTML如何转换到XHTML的信息，请参见附录B。

D.1.2　CSS

CSS有效性验证服务如图D-2所示，网址为http://jigsaw.w3.org/css-validator。

和HTML有效性验证一样，可以通过提交一个URL或上传一个样式表来进行有效性验证。要保证不提交同时包含HTML和CSS的文件，那会引起验证服务的混乱，足以导致自动有效性验证的失败。

验证CSS语法的另一种方法是把代码复制粘贴到页面底部的直接输入表单中。如果CSS在Web上是不可访问的或文件实际上是一个含CSS的HTML文件，那么这种方法可能最能满足您要求，同时也可能更快一些。



图D-2　W3C提供的CSS有效性验证服务

D.2　对元素的设置

在这一阶段，CSS语法是准确的，但没有更多的意义。

下面是直接对CSS本身进行操作。利用下列技术的一个或组合可以对CSS问题进行隔离。

D.2.1　把内外边距设置为零

浏览器所用的默认样式表给块级元素的内、外边距设置的是默认值。为了保证这些默

认值不影响设计，应该把块级元素的内、外边距设置为0。

用通配选择符可很容易地把内、外边距设置为0，设置起来也很快，如下所示：

```
* {
 margin: 0;
 padding: 0;
}
```

然后在样式表开始处添加这条CSS规则。这样其他具有内边距值、外边距值或两者都有的CSS规则，可以对已设置为0的内、外边距值进行重写。

说明：

如果需要用更健壮的CSS规则集来清除浏览器设置的默认值，可以试试Eric Meyer的样式表重置 (见 http://meyerweb.com/eric/thoughts/2007/05/01/reset-reloaded)。事实上，可以把CSS重置包含进来，作为正常工作流程的一部分。

注意页面设计的任何变化，然后做相应调整。

D.2.2 给边框和背景设置颜色

把正在分析的CSS规则突出显示，看看它们是不是导致问题的设计元素。一旦发现了导致问题的元素，就可以对问题进行修复。可以给边框或背景设置颜色以突出显示有问题的元素。

下面是一个示例：

```
#content #navigation {
  border: 1px solid red;
}
```

这个CSS规则在指定的块级元素周围创建一个红色边框，以便在页面设计中能更好地看到它。如果在设计中有太多红色以致不能注意到红色轮廓，则试试蓝色或绿色，或简单改变一下背景色，如下所示：

```
#content #navigation {
  background-color: green;
}
```

D.2.3 在属性值中添加变量

在找到导致问题的CSS之后，下一步就是调整这些属性的值。是在一个浏览器中内边

距太大吗？在另一个浏览器中字体是否太小？

修改属性值时，以非常大的值开始。例如，把内边距从25px改为2500px，看看设计的突变是否与预期的一致。

然后下一步是做小的调整。使用微小增量，例如，把字体大小从0.8em调整到0.81em。

D.2.4　捉谜藏

编写CSS规则的方式也可能导致问题。设置CSS，使某些属性及其值能由其孩子继承。例如，如果为body元素设置了字体属性，则body内的子元素也具有这些特征。

在构建CSS时，可能由于层叠、继承和优先级之间有冲突，编写的规则所继承的值不是我们所希望的。如果认为这是问题所在，可以从有问题的CSS规则把不必要的“属性/值”对注释掉，然后刷新页面看看变化情况。在下列代码段中，可在font-size声明块前添加一个字母x，使它不再显示。

```
#downloads h2 {
  font-family: "Myriad Pro", "Myriad Web", sans-serif;
  xfont-size: 1.4em;
  text-transform: uppercase;
}
```

D.2.5　再次进行验证

在排除问题的这一阶段，CSS可能被重写、修改或完全混合。要再一次进行有效性验证，保证没有遗漏什么东西。

D.3　寻求外界帮助

如果没有找到CSS问题的原因，还可以寻求外界的帮助。可利用下列资源查找问题或寻求帮助。

D.3.1　Web站点资源

下面提供一些重要资源信息。

D.3.2 positioniseverything.net

或许不是CSS的问题，而是由浏览器引起。在http://www.positioniseverything.net列出了现代浏览器的很多错误及解决方法。

D.3.3 Web Developer工具条

如果用Netscape 7及以上版本、Mozilla或Firefox等浏览器进行开发，可跑去(不是走)求助于Web Developer，这是由Chris Pederick开发的浏览器扩展，网址为http://www.chrispederick.com/work/firefox/webdeveloper。

该工具提供了大量对Web设计人员和CSS争论者有益的功能，是排除CSS错误不可缺少的工具。在本指南中提到的一些技巧，单击Web Developer工具条上的按钮就能实现，不必手工编辑代码。

D.3.4 Firebug

Firebug(见http://getfirebug.com)是另一个Firefox扩展，用户可用它检查Web文档的HTML、CSS和JavaScript组件。除了可以检查代码外，Firebug还允许用户即时编辑并实时看到他们对文档的影响。

D.3.5 邮件列表

下面提供一些关键邮件列表资源的信息。

D.3.6 CSS讨论组

如果还有其他不能令您满意的事情，可以向CSS讨论组中热心肠的人们求助。这是一个邮件列表，专门讨论CSS设计中遇到的问题。邮件列表中的人既有专业人士，也有初学者，因此您遇到的问题他们可能已经遇到过。

有关列表的更多信息和如何加入的说明，请参见http://www.css-discuss.org。

D.3.7 Babble List

为了满足高级Web设计要求，Babble List团体相互交流信息、共享资源、进行理论探讨、交流设计人员和开发人员的实践经验，包括CSS开发经验。总体目标是提高技巧、共享发布的新版本。

有关列表的更多信息和如何加入的说明，请参见http://www.babblelist.com。

Christopher Schmitt, Todd Dominey, et al

Professional CSS: Cascading Style Sheets for Web Design, 2nd Edition

EISBN: 978-0-470-17708-2

图书在版编目(CIP)数据

CSS Web设计高级教程(第2版)/(美)施密特(Schmitt, C.)，多米尼(Dominey, T.)等著；窦朝晖 译.

—北京：清华大学出版社，2009.7

书名原文：Professional CSS: Cascading Style Sheets for Web Design, 2nd Edition

ISBN 978-7-302-20311-7

Ⅰ.C… Ⅱ.①施… ②多… ③窦… Ⅲ.主页制作—软件工具，CSS—程序设计 Ⅳ.TP393.092

中国版本图书馆CIP数据核字(2009)第092684号

责任编辑：王 军 于 平

装帧设计：孔祥丰

责任校对：成凤进

责任印制：李红英

出版发行：清华大学出版社 **地 址**：北京清华大学学研大厦 A 座

http://www.tup.com.cn **邮 编**：100084

社 总 机：010-62770175 **邮 购**：010-62786544

投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn

质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 刷 者：北京市世界知识印刷厂

装 订 者：北京市密云县京文制本装订厂

经 销：全国新华书店

开 本：185×230 **印 张**：19 **字 数**：380 千字

版 次：2009 年 7 月第 1 版 **印 次**：2009 年 7 月第1 次印刷

定 价：68.00 元

PROGRAMMER TO PROGRAMMER™

CSS Web设计高级教程(第2版)

Professional CSS: Cascading Style Sheets for Web Design, 2nd Edition

本书内容丰富详实，注重Web开发的最佳实践，反映了自第1版发行以来CSS Web设计所发生的变化。每章都以现实中的Web站点为例，提供了很多便于理解的CSS技巧和技术，这些技巧和技术已在特定站点得到应用。各章演示了Web站点从开始到结束的设计过程，深入探讨了设计人员如何克服在站点开发过程中遇到的特殊挑战，并提供了解决问题的不同方法。

本书由浅入深地对基于Web标准的大规模专业级CSS Web站点进行了研究，对一些常见问题提供了易于理解的解决方法，对如何高效开发以CSS驱动的专业级Web设计给出了实用的方案。

本书主要内容

- 使用含CSS的XHTML的最佳实践
- 如何使博客在外观和感觉上焕然一新
- 一个拥有数百万用户的Web站点的设计细节
- 在Web站点中包含阴影、下拉菜单和嵌入式Flash内容的技术
- 解决浏览器兼容问题和开发功能性导航结构的技巧
- 通过CSS编码定制Web站点的方法
- 如何创建HTML Email模板和基本HTML表格布局，以及CSS在文中所起的作用
- 网格和布局对设计的重要性

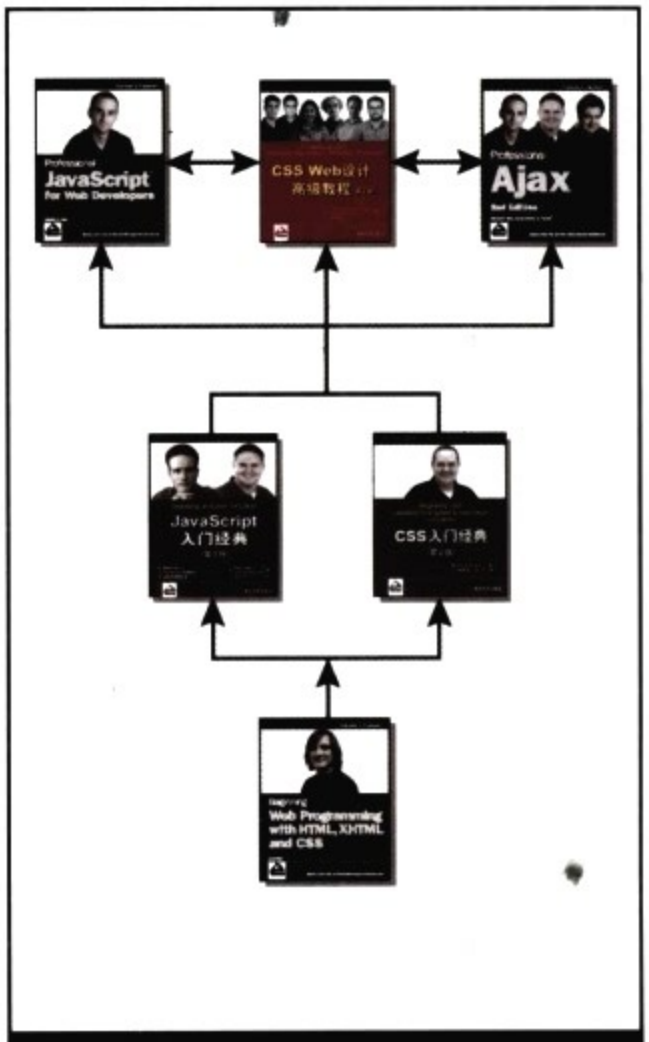
本书读者对象

本书针对希望透彻了解如何用CSS创建专业水平Web站点的Web开发人员。

本书源代码下载与技术支持

http://www.wrox.com

http://www.tupwk.com.cn/downpage



提升您的编程技能
完善您的职业生涯

Wrox Professional guides are planned and written by working programmers to meet the real-world needs of programmers, developers, and IT professionals. Focused and relevant, they address the issues technology professionals face every day. They provide examples, practical solutions, and expert education in new technologies, all designed to help programmers do a better job.

wrox™
An Imprint of WILEY

ISBN 978-7-302-20311-7

9 787302 203117 >

定价：68.00元

图书上架分类建议：Internet；网页设计